UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Instituto de Física y Matemáticas

Licenciatura en Matemáticas Aplicadas

Detección de Malware en sistemas Android usando aprendizaje máquina

TESIS

que para obtener el grado de

Licenciada en Matemáticas Aplicadas

presenta

Jasmín Martínez Espinosa

Director de tesis:

Dr. Tomás Pérez Becerra

Huajuapan de León, Oaxaca *Abril de 2024*

Dedicatoria

A mi madre Elia Concepción Espinosa López que sin su amor, paciencia y valentía, yo no estaría terminado esta etapa tan importante de mi vida.

Agradecimientos

Con este trabajo he avanzado tan sólo una pequeña parte de todo el camino que me queda por recorrer y quisiera agradecer a las personas que formaron parte de su elaboración.

A *mi madre:* usted tuvo la fuerza y valor para enfrentar su miedo a la soledad al dejarme ir a estudiar a una escuela que esta a 162 km de distancia de Villa de Etla, donde a pesar de todas las dificultades nunca dejo de apoyarme. De igual manera a mi *padre* que aunque ya no esta con vida sigo llevando en mente sus valores y enseñanzas.

A *mi tía Socorro:* pues no dudó en apoyar mi decisión de estudiar matemáticas aplicadas aunque esta idea no le agradara más que estudiar Físico-Matemáticas, que a pesar de todo y de ser la persona más ocupada del mundo que conozco siempre estuvo para mí, ante cualquier situación tuvo la mente fría para pensar en una solución.

A *mi Mamánes:* por darme sus palabras de aliento y apoyo cuando me veía decaída, por ese dinero que me daba a escondidas para mis dulces que aunque no lo destinaba a eso este me sacaban de apuros.

A *mis hermanos Román y Maximino* que en algún momento de mi estancia en la licenciatura me brindaron ayuda y por darme a mis cinco sobrinos Lluvia, Diego, Tanibeth, Elias y Felipe, que han sido y serán 5 motivos para no rendirme.

A *Roberto*, compañero de si no todos, por lo menos de un 70 % de los desvelos, sin tu paciencia para explicarme los conceptos que no entendía inmediatamente hubiera fallado en algunas pruebas, agradezco enormemente a la vida por haberte puesto en mi camino ya que fuiste y sigues siendo mi persona favorita para estudiar, para aprender tanto de temas académicos como de la vida, no dudes que te apoyaré en los proyectos que emprendas y cuando nos separemos por un intervalo de tiempo, aquí estaré esperándote mientras escucho "waiting for love" y guardo un "Hola compita" para ti.

A *mis amigos:* Diana Saraí pues a pesar de la distancia y que nos veamos cada cierto par de años, no fue impedimento para alentarnos en conseguir nuestros objetivos, espero con mucha ansiedad escuchar tus pato-aventuras que no han de ser pocas; a Daniela, aún recuerdo la primera ocasión que hablamos, esa tarde que nos encontramos a fuera de la biblioteca y nos fuimos juntas a la charla de bienvenida a los alumnos de nuevo ingreso, desde ese momento nos hicimos muy unidas, agradezco tu apoyo, comprensión y alegría; a Cecilia, Gamaliel, Héctor, José, Juan Felipe, Lorena y Ludwig, por esas risas interminables durante clases, horas libres o cuando decidíamos ir por un esquite simplemente para recordar anecdotas graciosas y hasta un tanto desastrosas.

A *Sra. Rosario Ojeda Alavéz:* ya que me brindo apoyo cuando no tenía con quien recurrir, por brindarme esa comprensión y cariño pues fue difícil llegar a un lugar donde no estaba mi familia.

A *mi director de tesis:* él Doctor Tomás por su paciencia, dedicación y compromiso con este trabajo, le estoy muy agradecida por brindarme la oportunidad de poder aprender bajo su tutela tanto una nueva área de las matemáticas (de la cual quedé prendada), como una actitud positiva, organización, pasión y mucha dedicación por la investigación; sin duda alguna esta experiencia ha sido un parte aguas en mi desarrollo académico y profesional.

A *mis revisores y sinodales:* por brindarme su tiempo en la revisión, además de aconsejarme en las correcciones, ya que estas me ayudaron a conocer conceptos que aun no comprendía.

A *mis profesores* los cuales dejaron una marca en mi memoria, aprendí de ustedes tanto académicamente como profesionalmente, de todos los profesores que tuve durante esta estancia en la universidad quiero agradecer especialmente a la Maestra Ana Delia que fue mi tutora en casi todos los años de la universidad, esas charlas en su cubículo donde me escuchaba y me daba consejos las tengo presentes no solo en la memoria si no también en el corazón.

Por último pero no por menos importante a mi profesor de matemáticas de la secundaria, el *profesor Liborio Lara Balois*, que si no es por usted yo no me hubiera enterado de la existencia de la Universidad Tecnológica de la Mixteca, de la licenciatura en Matemáticas Aplicadas, usted es de esos profesores que además de enseñar muy bien su materia, también deja huella en los corazones de sus alumnos; cuando me preguntan por que escogí estudiar esta carrera, sin titubear respondo que elegí esta profesión por un maestro de la secundaria que me enseño a querer las matemáticas.

Índice general

Introducción **IX**

1. Preliminares en Sistemas Android **1**

1.1. Marco conceptual 1

2. Gestión de la base de datos **7**

2.1. Características de la base de datos 8

2.1.1. Datos perdidos y faltantes 10

2.2. El Problema de clasificación 11

2.3. Variables y su representación 14

2.4. Análisis de correlación 16

3. Modelo de aprendizaje máquina supervisado **23**

3.1. Problemas de clasificación y regresión 23

3.2. Modelos de máquina de soporte vectorial 24

3.3. Maquina de soporte vectorial para selección de variables 27

3.4. Maquina de soporte vectorial para clasificación 32

3.4.1. Entrenamiento 33

3.4.2. Validación 35

3.4.3. Simulación 38

Conclusiones **41**

Índice alfabético **44**

Anexo 1 **47**

Anexo 2 **57**

Anexo 3 **61**

Anexo 4 **65**

Anexo 5 **69**

Anexo 6 **71**

Introducción

La tecnología digital ha formado una parte esencial de la vida cotidiana de cada individuo de la sociedad moderna desde hace algunos años, el uso de esta se vio aumentada cuando surgieron los dispositivos móviles. Algunas personas vieron su inicio, a otros les toco crecer junto con ella y el resto nacieron inmersos en esta era. Todos los dispositivos móviles están regidos por sistemas operativos, uno de ellos es el sistema *Android*, el cuál se encuentra no solo en celulares, sino también en tabletas, computadoras, televisores o relojes inteligentes. En los últimos años, Android se ha convertido en el sistema operativo móvil líder con el mayor porcentaje de cuota de mercado global que el resto. Se han vendido más de mil millones de dispositivos Android y tan solo en 2017 se estima que se han descargado 65 mil millones de aplicaciones solo desde Google Play ([14]). El crecimiento de la popularidad de Android y la proliferación de mercados de aplicaciones de terceros también lo han convertido en un objetivo popular del malware. En el 2016, McAfee informó que había más de 12 millones de muestras de malware para Android y que cada año se descubrían casi 2,5 millones de muestras nuevas ([9]). El malware de Android puede integrarse en una variedad de aplicaciones, como aplicaciones bancarias, aplicaciones de juegos, aplicaciones de estilo de vida y aplicaciones educativas.

El malware móvil es un software malicioso que se dirige a teléfonos móviles o Asistentes Digitales Personales (PDA) inalámbricos, provocando el colapso del sistema y la pérdida o fuga de información confidencial. A medida que los teléfonos inalámbricos y las redes

de PDA se han vuelto cada vez más comunes y han crecido en complejidad, se ha vuelto cada vez más difícil garantizar su seguridad contra ataques electrónicos en forma de virus u otro malware.

El malware pueden entonces comprometer la seguridad y la privacidad de las aplicaciones, al permitir el acceso no autorizado a información sensible a la privacidad, rootear dispositivos es decir que dota de los máximos privilegios posibles a un dispositivo móvil, evitando las limitaciones que impone el fabricante del dispositivo o la operadora que lo vendió, convertir dispositivos en bots controlados remotamente, entre otras acciones. Para introducir un contexto a esta investigación se darán respuesta a las siguientes preguntas:

- **¿Qué es Android?** Android es un sistema operativo diseñado para cualquier dispositivo móvil con pantalla táctil o sistemas multimedia, que van desde los celulares hasta los reproductores en algunos modelos de automóviles. Fue desarrollado por *Google* y basado en el Kernel de *Linux* y otros programas de código abierto. Android se ha convertido en el principal responsable de la popularización de muchos medios digitales por el hecho de facilitar el uso de una gran cantidad de aplicaciones de forma sencilla ([1]).

- **¿Cómo surgió Android?** Su historia comenzó en octubre de 2003, mucho antes de que se usará el término teléfono inteligente y varios años previos al anuncio de *Apple* de su primer *iPhone* e *iOS*. *Android Inc.* se fundó en Palo Alto, California. Sus cuatro fundadores son Rich Miner, Nick Sears, Chris White y Andy Rubin. Este último mencionó que la compañía iba a desarrollar *"dispositivos móviles inteligentes que sean más conscientes de la ubicación y las preferencias de su propietario"*.

  El siguiente capítulo en la historia de Android comenzó en 2005, cuando Google adquirió la empresa original. Rubin y otros miembros fundadores se quedaron con el objetivo de continuar desarrollando el sistema operativo (SO) con sus nuevos propietarios, se tomó la decisión de utilizar Linux como base, eso significaba que podría ofrecerse a fabricantes de teléfonos móviles de forma gratuita.

  Solo unos años después del lanzamiento del primer sistema operativo Android 1.0,

los teléfonos inteligentes con el nuevo SO estaban en todas partes, actualmente ha entrado en vigencia la versión 11, se invita a consultar la lectura [8] con el fin de conocer un poco más de historia sobre Android.

- **¿Que es un malware?** Malware o *"software malicioso"* es un término amplio que describe cualquier programa o código que es dañino en los sistemas. Es hostil, intrusivo e intencionadamente desagradable, intenta invadir, dañar o deshabilitar ordenadores, equipos informáticos, redes, tabletas y dispositivos móviles, a menudo asumiendo el control parcial de las operaciones de un dispositivo. Al igual que la gripe en el cuerpo, interfiere en el funcionamiento normal del aparato ([10]).

- **¿De donde provienen?** Se puede entrar en contacto con malware de diversas formas, en sitios de mucho tráfico como redes pornográficas o de juegos de azar, o al abrir los adjuntos de correos electrónicos sospechosos. En su mayoría, provienen de China y de Estados Unidos, mayores productores de software del mundo ([6]). Ocasionalmente, estos programas maliciosos pueden provenir de empresas reconocidas, disimulados en sus productos. Un caso famoso ha sido el del proceso *"Windows Telemetry"*, que está incluido en las nuevas versiones de Windows, y que es capaz de recoger información sobre los hábitos del usuario y enviarlos secretamente a Microsoft, utilizando para ello recursos del sistema ([6]).

- **¿Solo existe un tipo de malware?** Lo ideal sería que solo existiera un tipo, pero hay varios, y además estos evolucionan conforme lo hace la seguridad informática, siendo estos los más comunes ([10]):

  Adware. Es un software diseñado con el propósito de mostrar anuncios en pantalla, normalmente en un explorador. Suele recurrir a un método subrepticio: bien se hace pasar por legítimo o se adosa a otro programa para engañar al usuario e instalarse en su PC, tableta o dispositivo móvil.

  Spyware. Registra las actividades del usuario en el ordenador en secreto y sin permiso, luego se las comunica al autor del software.

Virus. Es un malware que se adjunta a otro programa y, cuando se ejecuta (normalmente sin que lo advierta el usuario), se replica modificando otros programas del ordenador e infectándolos con su propio código.

Gusanos. Son un tipo de malware similares a los virus, la diferencia de estos es que no es necesario que exista la intervención humana, es decir, se replica por sí solo con el fin de diseminarse por otros ordenadores en una red, normalmente provocando daños, destruyendo datos y archivos.

Troyano o caballo de Troya. Es uno de los tipos de malware más peligrosos. Normalmente, se presenta como algo útil lo que cual logra engañar al usuario. Una vez que está en el sistema, los atacantes que se ocultan tras el troyano obtienen acceso no autorizado al ordenador infectado; desde allí, roban información financiera o instalan amenazas.

Ransomware. Bloquea el acceso del usuario al dispositivo o cifra sus archivos y después lo forza a pagar un rescate para devolvérselos, se ha reconocido como el arma preferida de los delincuentes informáticos porque exige un pago rápido y provechoso en criptomoneda de difícil seguimiento. El código que subyace en éste es fácil de obtener a través de mercados ilegales en línea y defenderse contra él es complicado.

Rootkit. Proporciona al atacante privilegios de administrador en el sistema infectado. Se diseña de modo que permanezca oculto del usuario, de otro software y del dispositivo.

Registrador de pulsaciones de teclas. Graba todas las pulsaciones de teclas del consumidor, almacena la información recopilada y se la envía al atacante, quien busca datos confidenciales, como nombres de usuario, contraseñas o detalles de la tarjeta de crédito.

Minería de criptomonedas maliciosas. Denominada también minería fortuita o cryptojacking, es un malware cada vez más prevalente, instalado por un troyano. Permite que otras personas utilicen su ordenador con el fin de crear

criptodivisas como bitcoin o monero. Los programas maliciosos utilizan los recursos del dispositivo pero envían los coins obtenidos a sus propias cuentas, no a las del propietario del equipo. En pocas palabras, un programa de estos roba bienes con el fin de generar dinero.

Exploits. Aprovecha los errores y vulnerabilidades de un sistema para que el creador pueda asumir el control. Están vinculados, entre otras amenazas, a la publicidad maliciosa, que ataca a través de una página legítima que descarga contenido malicioso inadvertidamente desde un lugar peligroso. A continuación, el contenido dañino intenta instalarse en el ordenador tras una descarga involuntaria sin siquiera dar clic sobre él, todo lo que tiene que realizar la víctima es visitar alguna vez un sitio infectado.

Por otro lado, la ciencia de datos ha demostrado ser una herramienta eficaz para brindar seguridad en sistemas operativos, es necesario conocer un poco sobre esta área, pues brinda las herramientas para resolver el problema, así que **¿Qué es la Ciencia de Datos?**

La respuesta más popular que se puede encontrar en Internet es que: "Es la disciplina encargada de analizar grandes volúmenes de información, con la ayuda de la inteligencia artificial, para mejorar su manejo". Esta definición es bastante cercana a lo que esta área realiza, pero al ser una disciplina emergente no ha sido completamente definida, en el libro *"The Data Science Design Manual"* de Steven S. Skiena ([12]) se menciona que se encuentra en la intersección de la informática, la estadística y los problemas aplicados. De la primera, se desprende el aprendizaje máquina y las tecnologías de computación; de la segunda, una larga tradición de análisis exploratorio de datos, pruebas de significancia y técnicas de visualización; y de la tercera, la utilidad en los negocios, en la ciencia, en la economía y, en general, en cualquier área donde se requiera tomar decisiones y comprender un fenómeno.

Esta disciplina comienza en 1962, cuando John W. Tukey introdujo el término *"Ciencia de datos"* en su artículo *"The Future of Data Analysis"*, al explicar una evolución de la estadística matemática, ahí se definió como *"Procedimientos para analizar cifras, téc-*

nicas para interpretar las conclusiones de dichos procedimientos, formas de planificar la recopilación de los números, con los cuales se puede realizar su estudio más fácil, preciso o acertado, y toda la maquinaria y los resultados de las estadísticas matemáticas que se aplican al análisis de dato" ([15]).

En este trabajo se aplican técnicas de ciencia de datos y aprendizaje máquina para diseñar un programa que genere alertas mientras simula posibles ataques de malware en función de la susceptibilidad del sistema a los ataques de malware. Estos atentados son un tema de seguridad relevante debido a que la mayoría de las personas utilizan algún dispositivo móvil con Android. Ayudar a combartirlos es un problema de gran interés y es precisamente el planteamiento a resolver en esta investigación, la forma de hacerlo es con el diseño de un algoritmo basado en los modelos de *máquina de soporte vectorial*, que realicé predicciones sobre ataques simulados a este sistema operativo. Se realizan utilizando la base de datos *"Android Malware Dataset for Machine Learning"* de libre acceso, publicada por Suleiman Yerima y Sakir Sezer (se recomienda consultar el artículo [16] donde es utilizada y difundida por los creadores) disponible en la dirección:

`https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653`,

la cual cuenta con 215 variables que se analizan mediante herramientas estadísticas.
El procedimiento a seguir, y que es la motivación de la estructura de la tesis, es:

1. Capítulo 1: Se brinda un marco conceptual de la problemática y se realiza la exploración inicial de la base de datos, con la finalidad de determinar su estructura general, el tipo y clases de variables, todo desde la perspectiva funcional de los sistemas Android.

2. Capítulo 2: Está dedicado al análisis exploratorio, se plantea formalmente la problemática y se establece el lenguaje matemático requerido en el procesamiento posterior. Aquí se puede encontrar el análisis de los datos perdidos o faltantes, la

representación algebraica de las variables que componen la base y un estudio de correlación. En este capítulo se observa que se requiere un estudio de selección de variables significativas, que se realiza en el Capítulo 3.

3. Capitulo 3: En este capitulo se plantea el modelo de aprendizaje máquina basado en las máquinas de soporte vectorial. Resultado de las diversas pruebas realizadas, se determina que se usa máquina de soporte vectorial para la selección de variables y para la clasificación, por lo que se inicia con una sección en la que se exponen los problemas de clasificación y regresión, luego se exhiben los modelos de máquina de soporte vectorial con el propósito de comprender la idea detrás de su funcionamiento, a continuación se adapta el modelo de regresión, el cual realiza la selección de variables relevantes. Se finaliza con el modelo utilizado en la clasificación binaria, se entrena, valida y finalmente se somete a determinadas simulaciones. Este capítulo finaliza con una comparación con algunos otros modelos existentes en la literatura.

4. Anexos. Se adjunta el código utilizado. Se encuentran 6 anexos: en el primero se exploran las variables que conforman la base; en el segundo se encuentra el análisis de correlación; en el tercero se halla el análisis exploratorio de datos; en el cuarto se muestra el método de selección de variables; el quinto corresponde al entrenamiento y en el sexto se muestra el código de la simulación.

A manera de justificación, el desarrollo de esta investigación proporcionará una herramienta de apoyo a la protección de sistemas Android ante ataques de malware, beneficiando directamente a toda la población que posea algún dispositivo de este tipo, esto coadyuva a incrementar la seguridad de la información almacenada en equipos.

Capítulo 1

Preliminares en Sistemas Android

El sistema operativo Android es uno de los más populares en la actualidad, confluye de diversos procesos computacionales que son asignados a tareas específicas para formar en su totalidad al sistema. Debido a que se desea determinar la vulnerabilidad ante malware, no se pueden obviar a todos sus componentes. Saber cómo funciona y entender qué elementos específicos deben analizarse es necesario para realizar la selección de las características. Por esta razón, se inicia la investigación conceptualizando y definiendo los componentes esenciales que controlan este popular sistema operativo y son mostrados en este capítulo, posteriormente, se relaciona cada concepto con las variables de la base de entrenamiento del modelo.

1.1. Marco conceptual

Los siguientes conceptos son elementales para entender de forma práctica la problemática abordada en esta investigación:

- **Dispositivo:** Es un aparato que desarrolla determinadas acciones. Su nombre está vinculado a que está dispuesto para cumplir con su objetivo.
- **Firma:** Es un permiso que concede el usuario a una aplicación con el objeto de

que esta pueda brindarle el servicio que describe, por ejemplo, las aplicaciones que implementan servicios con privilegios. Los servicios *Autocompletar* y redes privadas virtuales (*VPN*) también usan permisos de firma.

- **Host:** Es cualquier dispositivo conectado a una red que puede proporcionar o solicitar servicios, recursos o información.
- **Host module:** Es el encargado de moderar la conexión a internet.
- **App:** Es una aplicación de software diseñada para funcionar en dispositivos móviles, como *smartphones* y tabletas.
- **API:** Interfaz de programación de aplicaciones.
- **API call signature:** Permite a una aplicación solicitar datos o servicios de otra aplicación. La mayoría de las aplicaciones web realizan regularmente llamadas a la API.
- **Commands signature:** Firmas que se realizan en cada comando para que se ejecute una aplicación. Dependiendo del comando se puede ejecutar inmediatamente o quedar a la espera de que sea firmado.
- **Intent:** Es un objeto de mensajería que se utiliza para solicitar una acción de otro componente de una app. Si bien las intent's (intenciones) facilitan la comunicación entre componentes de varias formas, existen tres casos de uso principales: iniciar una actividad, iniciar un servicio y transmitir un aviso.
- **Manifest Permission:** Especifica un permiso del sistema que el usuario debe otorgar para que la app funcione correctamente. El usuario otorga permisos cuando se instala la aplicación, estos no necesariamente son los mismos permisos para todas las app's, por ejemplo, cuando se quiere mandar una ubicación en WhatsApp este solicita un permiso para poder acceder a la ubicación, a diferencia de la aplicación de llamada que te solicita el permiso para acceder a tus contactos.

Dentro de cada una de los anteriores componentes se encuentran una serie de instrucciones, algunas especificas son las mostradas a continuación.

- **API**
  - **AttachInterface**: Conecta una interfaz de red a una base de datos mediante la línea de comandos.
  - **Android.os.Binder**: También llamado Binder. Crea una carpeta donde se guardan datos del usuario de Android que el servicio después llama para poder interactuar con él.
  - **IBinder:** Es una subcarpeta de Binder que se encarga de mandar la información al usuario. Binder es el creador de las carpetas y IBinder es el mensajero.
  - **ServiceConnection**: Supervisa la conexión con el servicio.
  - **BindService**: Es un servicio enlazado que permite que los componentes de la aplicación se enlacen al dispositivo Android, con el cual crean una conexión a largo plazo.
  - **OnServiceConnected**: Cuando el sistema Android crea la conexión entre el cliente y el servicio llama a onServiceConnected.
  - **Transact**: Permiten enviar una llamada a un objeto IBinder y recibir una llamada entrante a un objeto Binder, respectivamente, como ejemplo, cuando el servicio que se esté ocupando necesite utilizar un Binder necesita un transact para poder transportar esta carpeta.
- **Commands signature**
  - **/system/bin**: Todos los binarios construidos como parte del código de Android (llamado AOSP), la cual pertenece al directorio principal de Android para almacenar componentes inmutables generados por la compilación del AOSP.
  - **/system/app**: Es donde se guardan todas las aplicaciones pre-instalas del sistema operativo.

- **mount**: La opción mount está presente en el modo de recuperación. Es un medio para modificar cualquier cosa que desee en un sistema o partición de datos. Este comando ayuda a ver, analizar y editar todos los archivos de datos en la partición de datos o del sistema.

- **Intent**

  - **android.intent.action.BOOT_COMPLETED**: Es una acción de difusión que se emite una vez, después de que el usuario haya terminado el arranque. Se puede utilizar para realizar una inicialización específica de una aplicación, como la instalación de alarmas.
  - **android.intent.action.PACKAGE_REPLACED**: Es una acción de difusión. Reemplaza una versión existente de un paquete que se instaló anteriormente. Los datos contienen el nombre del paquete.
  - **android.intent.action.SEND_MULTIPLE**: La aplicación puede procesar más de un elemento compartido.
  - **android.intent.action.BATTERY_OKAY**: Acción de transmisión, la cual indica que la batería ahora está bien después de estar baja. Esto se enviará una vez que la batería haya vuelto a un estado aceptable.

- **Manifest Permission**

  - **SEND_SMS**: Envía mensajes de texto, por ejemplo, la app de mensaje predeterminada.
  - **READ_PHONE_STATE**: Lee el estado de teléfono, es decir, permite a la aplicación saber tu número de teléfono, la información actual de la red móvil, el estado de las llamadas en curso, entre otros.
  - **GET_ACCOUNTS**: Permite acceder a la lista de cuentas.
  - **RECEIVE_SMS**: Permite recibir mensajes de texto.

Con los conceptos anteriores es posible describir las principales variables que influyen en que un sistema sea susceptible a ataques de Malware, tales variables se describen y se analizan a detalle en el siguiente capítulo.

Capítulo 2

Gestión de la base de datos

El proyecto *DREBIN* fue creado debido a la necesidad de incrementar la seguridad de la plataforma Android contra aplicaciones maliciosas y se compone por un método de análisis estadístico para detección de malware en Android. Daniel Arp y sus colaboradores, en la University of Göttingen en Alemania, desarrollaron dicho trabajo, dado a conocer mediante el artículo [3]. El objetivo principal de Drebin es reunir la mayor cantidad posible de características (información sobre las aplicaciones) de una aplicación. Estas características están introducidas en matrices. Dicho proyecto evalúa 123,453 aplicaciones y 5,560 muestras de malware. Se realiza sobre cinco populares smartphones y el análisis lo realiza en 10 segundos en promedio.

En este mismo sentido, Yajin Zhou y Xuxian Jiang, del Department of Computer Science de la North Carolina State University en Estados Unidos, coleccionan más de 1,200 muestras de malware en un proyecto que denominan *Android Malgenome project* y lo dan a conocer mediante la referencia [18].

La base que se utiliza en la presente investigación, y con la cuál se entrena el modelo de aprendizaje máquina, es un subconjunto de los datos obtenidos en los proyectos DREBIN, Android Malgenome project y una colección de muestras proporcionadas por *Intel Security*, formalmente por *McAfee*. Fue creada por Suleiman Y. Yerima y Sakir Sezer, ambos miembros del Institute of Electrical and Electronics Engineers (IEEE), y se dio a

conocer mediante el artículo "DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection" ([17]) de IEEE Transactions on Cybernetics. En esta investigación se denomina "Android Malware Dataset for Machine Learning" (brevemente AMDML) y se ha utilizado para desarrollar una serie de algoritmos mediante la combinación de dos clasificadores para la detección de malware de Android, por ejemplo en [17].

En este capítulo se explican las componentes de la base AMDML y se realiza el análisis exploratorio con el objetivo de obtener conclusiones parciales sobre su estructura, su procesamiento se realiza en un Notebook de *Colab* con las librerías *Pandas*, *Numpy*, *Seaborn*, *Matplotlib* y *SciKitLearn*, que permiten realizar cálculos de álgebra, manejo de datos, construir tablas y entrenar modelos de aprendizaje máquina.

2.1. Características de la base de datos

La base de datos contiene 15,035 registros (filas) y 216 características o variables (columnas), en el Cuadro 2.1.1 se muestran algunas de estas variables, de las cuales el 53 % pertenecen al tipo *Manifest Permission*, 33 % al de *API call signature* y un 14 % son de diversas clases. La primera etapa es la identificación de los tipos de datos disponibles para el análisis; que se clasifican dentro de alguna de las dos tipologías siguientes:

- **Datos orientados a la no dependencia:** Se refiere a tipos de datos simples, como datos multidimensionales o datos de texto, este tipo de datos son los más comúnmente encontrados. En este caso, las variables no tienen dependencias especificadas entre los elementos de datos o los atributos. Un ejemplo es un conjunto de datos demográficos sobre personas que contienen su edad, sexo y código postal. En el caso de la base AMDML es la variable *class* que tiene las clasificaciones de B y S.

- **Datos orientados a la dependencia:** En estos casos pueden existir relaciones implícitas o explícitas entre elementos de datos. Por ejemplo, un conjunto de datos de una red social contiene un conjunto de vértices (elementos de datos) que están conec-

tados entre sí por un conjunto de bordes (relaciones). Para el caso de la base de datos AMDML, las 215 variables restantes son de esta clase, por ejemplo, *transact, onServiceComnnected, SEND_ SMS y android.intent.action.BOOT_ COMPLETED.*

Dadas las descripciones anteriores se afirma que la base de datos AMDML está conformada por datos orientados a la no dependencia, esto debido a que cada dispositivo tiene un uso distinto, por ejemplo, un estudiante y un ama de casa no utilizan de la misma manera la función de cámara.

Además, la base se considerará conformada por datos multidimensionales en el sentido de la Definición 2.1.1 que a continuación se formula.

Definición 2.1.1. (Datos multidimensionales) Un conjunto de datos multidimensionales D es un conjunto de n vectores de características, $\overline{x_1}, ..., \overline{x_n}$ tal que cada variable $\overline{x_i}$ contiene un conjunto de características d denotadas por $(x_{1i}, ..., x_{di})$.

De esta forma, de ahora en adelante la base AMDML se denota por D y se considera como el siguiente arreglo:

	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	$\cdots$	$\overline{x_n}$	$\overline{y}$
$\overline{f_1}$	x_{11}	x_{12}	x_{13}	$\cdots$	x_{1n}	y_1
$\overline{f_2}$	x_{21}	x_{22}	x_{23}	$\cdots$	x_{2n}	y_2
$\overline{f_3}$	x_{31}	x_{32}	x_{33}	$\cdots$	x_{3n}	y_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\overline{f_d}$	x_{d1}	x_{d2}	x_{d3}	$\cdots$	x_{dn}	y_n

Específicamente, en la Tabla 2.1.2 se muestran los primeros cinco registros de la base D junto con algunas características. Se observa que se compone de 15,035 registros con 216 variables, esto es, $n = 15,035$ y $d = 216$.

La base de datos D se encuentra constituida por dos tipos diferentes de datos, el primero de ellos es:

- **Datos multidimensionales cuantitativos.** Se componen de valores numéricos como entradas y se conocen como continuo, numérico o cuantitativo. Los datos en los

que todos los campos son cuantitativos también se refieren como datos cuantitativos o datos numéricos. Así, cuando cada valor de x_i^d en la Definición 2.1.1 es cuantitativo, el conjunto de datos correspondiente se denomina multidimensional cuantitativo de datos, siendo estas las primeras 215 variables.

- **Datos de atributos categóricos y mixtos.** Los atributos como género, raza y código postal tienen valores discretos sin un ordenamiento natural entre ellos. Si cada valor de x_i^j en la Definición 2.1.1 es categórico, entonces dichos datos se conocen como datos de valores discretos no ordenados o categórica. En el caso de datos de atributos mixtos, hay una combinación de datos categóricos y atributos numéricos. En la base utilizada se encuentra la variable *class* que es de este tipo, esta variable tiene como atributos las letras B y S, que en lo sucesivo se consideran como -1 y 1, respectivamente.

En la tabla 2.1.1 se muestran algunos de los datos multidimensionales cuantitativos y en la 2.1.3 se aprecia la variable *class*, la cual es categórica.

2.1.1. Datos perdidos y faltantes

Los "datos faltantes" generalmente significa que son valores no disponibles que serían útiles o significativos para el análisis de los resultados. Aunque la ausencia puede ocurrir de forma aleatoria, el patrón de datos faltantes que describe las ubicaciones de los valores faltantes en relación con los observados puede proporcionar información útil sobre el proceso que conduce a la falta de datos y, por consiguiente, ayudar a formular algoritmos de solución para esta clase de problemas. Formalmente, en la Definición 2.1.2 se establece de manera abstracta a este tipo de datos vacíos.

Definición 2.1.2. (Patrón de datos faltantes) Sea $D = \{x_{ij}\}$, $i = 1, ..., n$, $j = 1, ..., d$, un conjunto de datos rectangular $n \times d$ sin valores faltantes (es decir, conjunto de datos completo o completamente observado). Su i-ésima fila es $\bar{f}_i = (x_{i1}, x_{i2}, ..., x_{id})$, donde x_{ij} es el valor de la variable $\bar{x}_j$, $(j = 1, ..., d)$, para el sujeto i. Cuando se producen datos

faltantes, algunos de los x_{ij}'s no se observan. Para describir esto, se define la matriz de indicadores de respuesta $R = r_{ij}$, donde $r_{ij} = 1$, si x_{ij} es observado, y $r_{ij} = 0$, si x_{ij} está perdido.

Para este caso, la matriz indicadora R resultó estar completa por 1's, esto es, no existen valores perdidos o faltantes (véase el diagrama de calor, enseñado en la Figura 2.1). En el caso de que sí se tuvieran datos faltantes, se utilizan tres clases de técnicas para manejar las entradas que faltan:

1. Cualquier registro de datos que contenga una entrada faltante puede eliminarse por completo. Sin embargo, este enfoque puede no ser práctico cuando la mayoría de los registros contienen entradas faltantes.

2. Los valores faltantes pueden ser estimados o imputados. No obstante, los errores creados por el proceso de imputación pueden afectar los resultados del algoritmo de minería de datos.

3. La fase analítica está diseñada de tal manera que puede trabajar con valores faltantes. Muchos métodos de minería de datos están diseñados para funcionar de manera sólida con valores faltantes. Este enfoque suele ser el más deseable porque evita los sesgos adicionales inherentes al proceso de imputación.

El problema de estimar las entradas faltantes está directamente relacionado con el problema de clasificación, en el cuál el algoritmo clasificador se utiliza para estimar el valor de los datos perdidos o faltantes. Este tema se deja abierto para posteriores investigaciones.

2.2. El Problema de clasificación

El objetivo de este trabajo es realizar una clasificación de la base de datos en dos tipos: sin riesgo de ataque de Malware o susceptible a ataque. En esta sección se plantea formalmente el problema de clasificación.

Figura 2.1: Diagrama de calor de datos faltantes. Los datos existentes son mostrados de color obscuro mientras que los perdidos o faltantes aparecen de color claro. Se observa que no existen datos de este tipo.

Este problema se estudia desde la técnica conocida como minería de datos, en este caso, la variable objetivo se representa por un dato binario -1 o 1 y se hace referencia a esta característica como la etiqueta de clase. Por lo tanto, este problema requiere de un programa supervisado, en donde estas relaciones se aprenden directamente de las características restantes en los datos. Los datos que se utilizan para aprender estas relaciones son denominados *datos de entrenamiento* y se denotan como D_{Train}, que es una base formada por registros d-dimensionales pertenecientes a la base D. El modelo de aprendizaje se utiliza para determinar las etiquetas de clase estimadas para los registros nuevos que se introducen, es decir, en estos se requieren la etiqueta.

En [2] el autor define el problema de clasificación, la siguiente es una adaptación:

Definición 2.2.1 (**Problema de clasificación de datos**)**.** Dada una $m \times d$ matriz de datos de entrenamiento D, con $m < n$ vectores fila d-dimensionales de la matriz de datos $\mathcal{D}$, y un conjunto de valores etiquetas de clase $\{-1, 1\}$ asociados con cada uno de los m renglones en D(entradas de $\mathcal{D}$), crear un modelo de entrenamiento $\mathcal{M}$, el cual pueda ser utilizado para predecir la etiqueta de clase de una entrada d-dimensional $\bar{x} \notin \mathcal{D}$.

El registro cuya etiqueta de clase se desconoce se denomina *registro de prueba*. Para el presente caso, la supervisión a partir de los datos de entrenamiento deberá reconocer los dos grupos de registros "similares", esto es, el modelo deberá aprender la estructura del conjunto de datos de entrenamiento. El modelo después del aprendizaje se denomina *modelo de entrenamiento*. Los registros de datos que el programa no ha visto anteriormente y que deben clasificarse se denominan *conjunto de datos de prueba*. El problema de aprendizaje supervisado utiliza un conjunto de datos de prueba para aprender la estructura de los grupos, al igual que un maestro supervisa a sus alumnos hacia un objetivo específico.

En resumen y de manera intuitiva, el problema de clasificación puede formularse como:

Dado un conjunto entrada formado por datos de entrenamiento, cada uno de los cuales está asociado con una etiqueta de clase, determinar la etiqueta de clase de una o más instancias de prueba nunca antes vistas.

El algoritmo de clasificación propuesto tiene dos fases:

Fase de entrenamiento: En esta fase se construye un modelo entrenado a partir de las instancia D_{Train}. Específicamente, es un modelo matemático que tiene por entrada a los grupos etiquetados del conjunto de datos de entrenamiento.

Fase de prueba: En esta fase, el modelo entrenado se emplea para determinar la etiqueta de clase (o identificador de grupo) de una o más instancias no antes vistas por el programa y cuyas etiquetas son conocidas solo por el programador. Estos registros son los elementos en D que no pertenecen a D_{Train}, y forman una nueva instancia llamada D_{Test}, esto es, $D_{Test} = D \setminus D_{Train}$.

La salida del algoritmo de clasificación es la predicción de etiquetas; es decir, presenta una etiqueta -1 o 1 para cada instancia de prueba. Cuando el conjunto de datos de entrenamiento D_{Train} es pequeño, el rendimiento del modelo de clasificación puede ser deficiente, esto es, el modelo, puede describir las características aleatorias específicas del conjunto de datos de entrenamiento pero no puede generalizarse a la estructura de grupo de instancias de prueba nunca antes vistas D_{Test}; en otras palabras, está puede predecir con precisión las etiquetas de las instancias utilizadas para construirlos, pero funcionar mal en instancias de prueba nuevas. Este fenómeno se conoce como sobre-ajuste y para evitarlo se utiliza $\frac{2}{3}$ partes de la base $\mathcal{D}$ como el conjunto de entrenamiento D_{Train} y el resto para las pruebas D_{Test}.

2.3. Variables y su representación

Los vectores columna n-dimensionales de la base D, que contienen variables numéricas, pueden ser descritos mediante las medidas de tendencia central:

- **Media:** $\overline{X_i} = \dfrac{\sum_{i=1}^{d} x_i^j}{d}$.
- **Varianza:** $\sigma_i^2 = \dfrac{\sum_{i=1}^{d} (x_i^j - \overline{X_i})^2}{d}$.
- **Desviación estándar:** $\sigma_i = \sqrt{\sigma_i^2}$.

Del Cuadro 2.3.1 se puede extraer la información de que algunas de las variables en la matriz de datos se conforman casi en la misma cantidad de 1's y 0's, puesto que la media está alrededor de 0.5; además, en una inspección más profunda, la mayoría se conforman por solo 0's y 1's, véase la gráfica superior derecha en la figura 2.2, por lo que se tienen diversas variables binarias. También se observa que existen variables que, relativamente, tienen una mayor cantidad de valores 0's y una mínima o casi nula cantidad de 1's, comúnmente, cuando esto ocurre, la variable se considera constante y es probable que se elimine, sin embargo, en este estudio, esto se realiza en la Sección 3.3 mediante la selección de variables significativas .

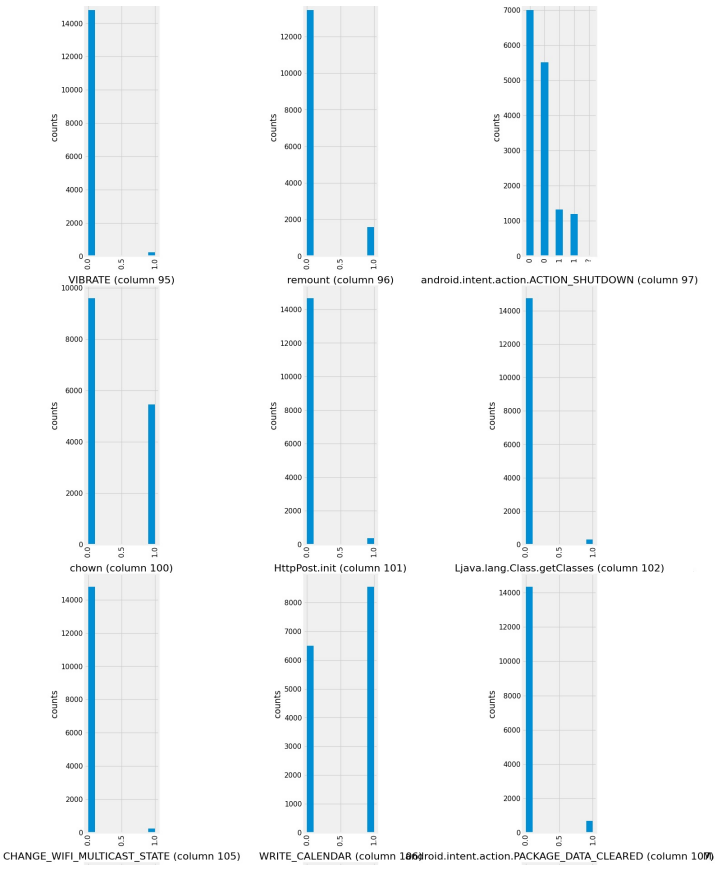

Figura 2.2: Distribución de frecuencias de algunas variables.

Por otro lado, recordemos que la variable *class* que hace referencia a aquellos registros que han sido atacados por un Malware y aquellos que no lo han sido. Los valores que toma

esta variable son B y S, representando a "Bening" por benigno y "S" por susceptible a los Malware. En la Figura 2.3 se observa que se distribuyen 9,476 registros benignos y 5,560 con Malware. Esta distribución de frecuencias es ideal para lograr un buen desempeño en el entrenamiento de los modelos de aprendizaje máquina.

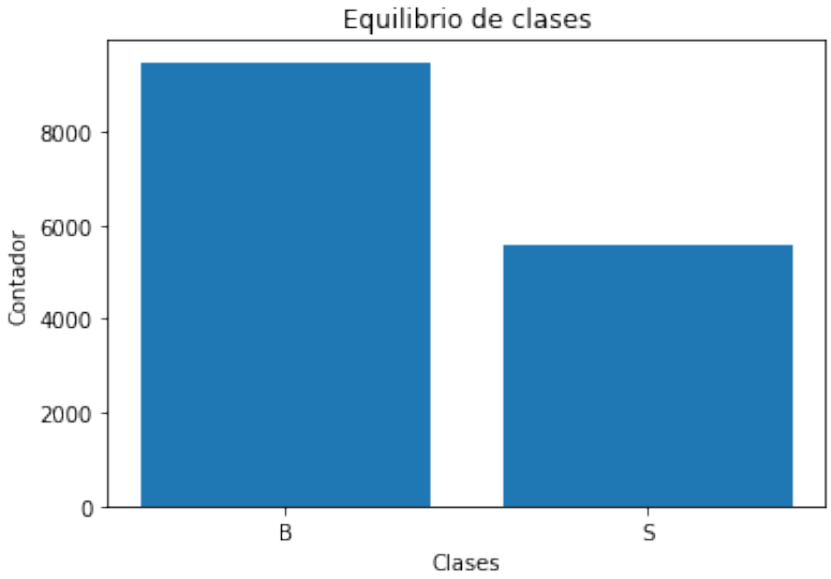

Figura 2.3: Distribución de los registros para la variable *class*.

Esta variable se codifica mediante la transformación $B \mapsto -1$ y $S \mapsto 1$, de esta forma, la problemática se traduce como la predicción de la etiqueta -1 o 1 para nuevos registros.

2.4. Análisis de correlación

Para esta etapa se requiere determinar si es viable la selección de variables, esto es, si existe alguna correlación entre ellas. Para este caso, se dice que las variables X e Y están correlacionadas cuando el valor de X tiene cierto poder predictivo sobre el valor de Y. El coeficiente de correlación $\rho(X, Y)$ es un estadístico que mide el grado de Y en función de X y viceversa. El coeficiente de correlación toma valores en el intervalo $[-1, 1]$, donde 1 significa totalmente correlacionado y 0 sin relación o que X y Y son variables independientes. Las correlaciones negativas implican que las variables están anticorrelacionadas,

lo que significa que cuando X sube, Y baja. El análisis de correlación empleado en esta tesis es realizado mediante el *coeficiente de correlación de Pearson.*

El coeficiente de correlación de Pearson entre un par de variables aleatorias $X = (x_1, x_2, \ldots, x_n)$ e $Y = (y_1, y_2, \ldots, y_n)$ se define como

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 * \sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

donde $\bar{x}$ e $\bar{y}$ representan la media respectiva de cada variable.

Algunos de los valores de los coeficientes de correlación, resultados del cálculo de ρ de la combinación bivariada entre todas las variables, son ilustrados en la Figura 2.4.

	transact	onServiceConnected	bindService	attachInterface	ServiceConnection	android.os.Binder	SEND_SMS
transact	1.000000	0.810812	0.816442	0.963364	0.813181	0.881938	-0.274731
onServiceConnected	0.810812	1.000000	0.983198	0.783617	0.984796	0.842693	-0.324517
bindService	0.816442	0.983198	1.000000	0.791208	0.994622	0.848719	-0.321665
attachInterface	0.963364	0.783617	0.791208	1.000000	0.787798	0.861441	-0.263826
ServiceConnection	0.813181	0.984796	0.994622	0.787798	1.000000	0.848134	-0.322458
...	...	...	...	...	...	...	...
ACCESS_FINE_LOCATION	0.121453	0.126678	0.128311	0.119865	0.132108	0.102486	-0.008198
SET_WALLPAPER_HINTS	-0.008323	-0.018780	-0.017788	-0.011382	-0.018360	-0.021553	0.016163
SET_PREFERRED_APPLICATIONS	-0.005148	-0.002141	-0.001477	-0.002906	-0.001860	-0.015285	-0.011377
WRITE_SECURE_SETTINGS	-0.000828	0.007813	0.008242	0.005000	0.006604	0.000550	0.108017
class	-0.570235	-0.561192	-0.558963	-0.551578	-0.557725	-0.553786	0.545650

Figura 2.4: Algunos valores del coeficiente de correlación para cada par de variables.

La representación gráfica de las correlaciones entre algunas parejas de variables se ilustran en el mapa de calor de la Figura 2.5. El mapa completo se encuentra en el Anexo 2 de esta tesis. En el mapa se puede apreciar que el color claro representa una fuerte correlación, por el contrario, si los colores son más obscuros e inclusive negro, su correlación es débil, se puede observar en el mapa completo del anexo que las correlaciones de la mayoría de las variables son débiles.

De lo anterior se concluye que existen algunas variables que están fuertemente correlacionadas, las que presentan una correlación superior a 0.5, tanto positiva como negativa, con la variable *class* son:

- Manifestación de permiso:

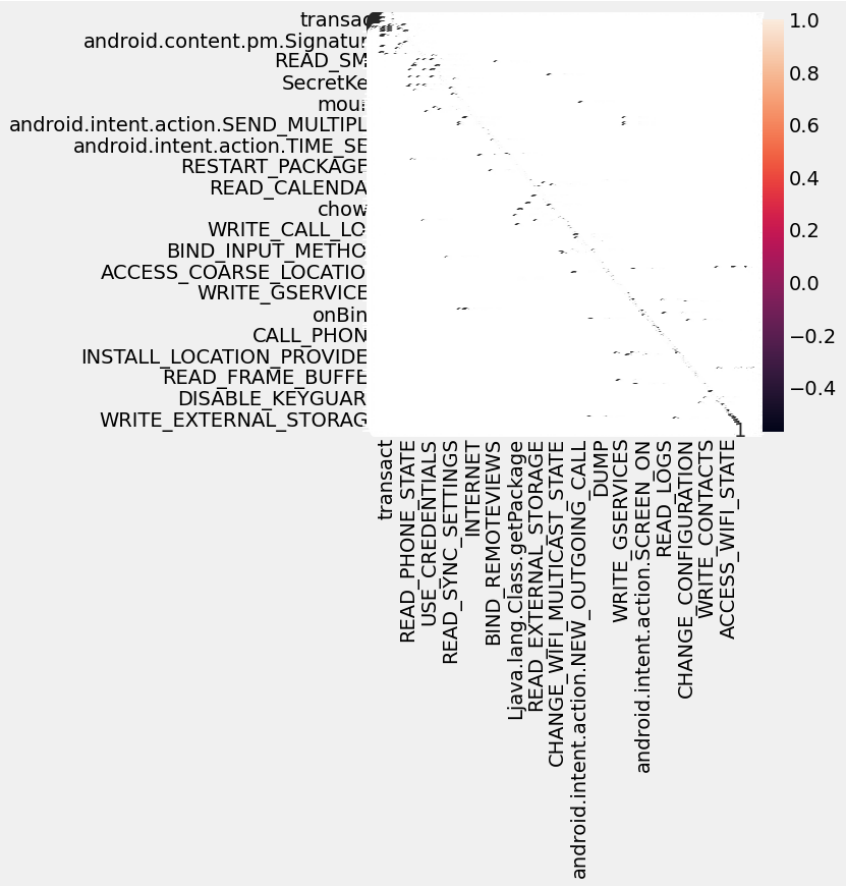

Figura 2.5: Mapa de calor de las correlaciones entre algunas variables.

1) **SEND_SMS**.

- API:

2) **attachInterface**
3) **android.os.Binder**
4) **ServiceConnection**
5) **bindService**
6) **onServiceConnected**
7) **transact**

Estas correlaciones se pueden observar en las figuras 2.6 y 2.7

	SEND_SMS	attachInterface	android.os.Binder	ServiceConnection	bindService	onServiceConnected	transact	class
SEND_SMS	1.000000	-0.263826	-0.304738	-0.322458	-0.321665	-0.324517	-0.274731	0.545650
attachInterface	-0.263826	1.000000	0.861441	0.787798	0.791208	0.783617	0.963364	-0.551578
android.os.Binder	-0.304738	0.861441	1.000000	0.848134	0.848719	0.842693	0.881938	-0.553786
ServiceConnection	-0.322458	0.787798	0.848134	1.000000	0.994622	0.984796	0.813181	-0.557725
bindService	-0.321665	0.791208	0.848719	0.994622	1.000000	0.983198	0.816442	-0.558963
onServiceConnected	-0.324517	0.783617	0.842693	0.984796	0.983198	1.000000	0.810812	-0.561192
transact	-0.274731	0.963364	0.881938	0.813181	0.816442	0.810812	1.000000	-0.570235
class	0.545650	-0.551578	-0.553786	-0.557725	-0.558963	-0.561192	-0.570235	1.000000

Figura 2.6: Variables con correlación positiva o negativa con la variable *class* mayor a 0.5.

El análisis de correlación da paso a un primer acercamiento sobre la existencia de variables con mayor influencia con la variable objetivo *class*, que junto con la verificación de no existencia de datos faltantes, muestran el camino a seguir para la selección de variables y su posterior implementación del modelo de clasificación. Estos procesos se establecerán en el siguiente capítulo.

Figura 2.7: Mapa de calor de variables con correlación positiva o negativa con la variable *class* mayor a 0.5.

Variable	**Tipo**
transact	API call signature
onServiceConnected	
bindService	
attachInterface	
ServiceConnection	
android.os.Binder	
Ljava.lang.Class.getCanonicalName	
Ljava.lang.Class.getMethods	
Ljava.lang.Class.cast	
Ljava.net.URLDDecoder	
android.content.pm.Signature	
android.telephony.SmsManager	
SEND_SMS	Manifest Permission
READ_PHONE_STATE	
GET_ACCOUNTS	
RECEIVE_SMS	
READ_SMS	
USE_CREDENTIALS	
MANAGE_ACCOUNTS	
WRITE_SMS	
WRITE_HISTORY_BOOKMARKS	
android.intent.action.BOOT_COMPLETED	Intent
android.intent.action.PACKAGE_REPLACED	
android.intent.action.SEND_MULTIPLE	
android.intent.action.TIME_SET	
android.intent.action.PACKAGE_REMOVED	

Cuadro 2.1.1: Datos multidimensionales cuantitativos de la base D.

	transact	bindService	attachInterface	SEND_MS	···	class
1	0	0	0	1	···	S
2	0	0	0	1	···	S
3	0	0	0	1	···	S
4	0	0	0	0	···	S
5	0	0	0	0	···	S

Cuadro 2.1.2: Primeros cinco registros de la base D.

Variable	Tipo
class	B=Benign=−1; S=Malware=1

Cuadro 2.1.3: Variable categórica *class* y su asignación binaria.

	transact	bindService	attachInterface	SEND_MS
Cantidad	15036	15036	15036	15036
Media	0.426443	0.442671	0.413208	0.236632
Desv. est.	0.494576	0.496719	0.492426	0.425029
Min	0	0	0	0
25 %	0	0	0	0
50 %	0	0	0	0
75 %	1	1	1	0
Max	1	1	1	1

Cuadro 2.3.1: Medidas de tendencia central de algunas variables numéricas.

Capítulo 3

Modelo de aprendizaje máquina supervisado

La existencia de una alta correlación en las variables, tanto positiva como negativa, da cabida a la idea de que se puede reducir el número de variables, por lo que en este capítulo se aborda la teoría detrás de la selección de variables significativas. Así mismo, se aborda este mismo tipo de modelos para la clasificación de variables y se muestra el proceso de entrenamiento.

Tanto para la selección de variables como para la clasificación, se utiliza un modelo de aprendizaje supervisado basado en las máquinas de soporte vectorial y un conjunto de datos etiquetados para entrenarlos, se realizará la medición de la precisión del algoritmo clasificatorio a través de las métricas de desempeño. Por lo anterior, el capítulo inicia con una breve descripción de los problemas de clasificación y regresión y la idea detrás de este tipo de modelos.

3.1. Problemas de clasificación y regresión

El aprendizaje supervisado puede clasificarse en dos tipos de problemas durante la minería de datos:

- **Clasificación:** La clasificación supervisada es el proceso que consiste en asignar

a una clase a cada objeto (también llamados vectores) dentro de un conjunto, en otras palabras reconoce entidades específicas dentro del conjunto de datos e intenta sacar algunas conclusiones sobre cómo esas entidades deben etiquetarse o definirse. El algoritmo utiliza un conjunto de objetos como entrenamiento y el objetivo es utilizarlo para clasificar nuevos objetos. El problema de clasificar objetos en clases surge en diversos campos, como las compañías de seguros (para diferenciar a los malos clientes de los buenos), la medicina (para determinar si un tumor es benigno o maligno), la química o en procesamiento de imágenes.

Existen diversos algoritmos de clasificación, algunos de ellos son máquinas de vectores de soporte (SVM), árboles de decisión, k vecinos más cercanos y bosques aleatorios.

- **Regresión:** Se utiliza para comprender la relación entre variables dependientes e independientes. Se utiliza comúnmente para hacer proyecciones, como los ingresos por ventas de un negocio determinado. Algunos algoritmos populares son regresión lineal, regresión logística y regresión polinomial.

En esta investigación se empleará un modelo de regresión para realizar la selección de variables (Sección 3.3). Así mismo, se utilizará uno de clasificación supervisada (Sección 3.4), puesto que se desea conocer cuando un dispositivo con sistema operativo Android es vulnerable a un ataque de Malware, ambos basados en las Maquinas de Soporte Vectorial.

3.2. Modelos de máquina de soporte vectorial

Una máquina de soporte vectorial (Support Vector Machines o SVM) es un popular modelo de aprendizaje supervisado desarrollado por Cortés y Vapnik en 1995 y aparece por primera vez en [7]. Este tipo de modelos se basan en construir un hiperplano donde la distancia entre dos clases es máxima.

Bajo ciertas condiciones, la SVM puede encontrar el mínimo global de la función objetivo, con lo que transforma problemas de predicción en problemas de optimización

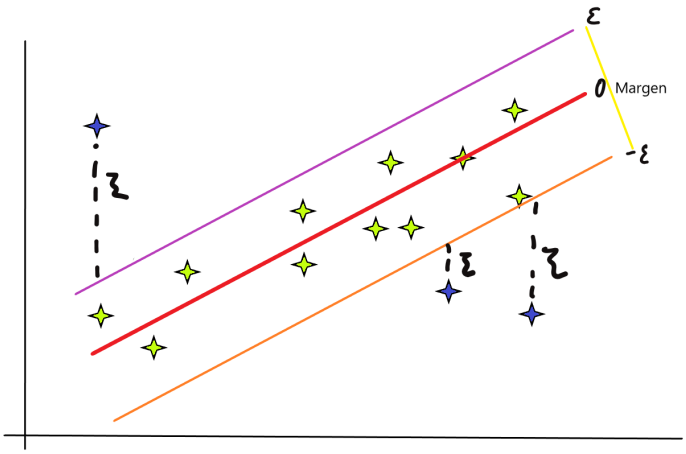

Figura 3.1: En rojo: función lineal óptima. Los puntos azules son considerados errores.

convexa. Supera varios inconvenientes de las redes neuronales artificiales, incluido el sobreajuste y la convergencia a mínimos locales ([4]).

Como se ha mencionado anteriormente, en esta investigación se utilizan los modelos de máquina de soporte vectorial para realizar tanto la selección de variables como la clasificación. Por lo que se expone en cada contexto iniciando con el primero.

Para realizar la selección de variables se adapta un modelo de máquina de soporte vectorial para regresión. En los problemas de regresión lineal el objetivo es ajustar una línea a una nube de puntos que minimice las distancias entre ellas, es decir, se desea hallar una función

$$\phi(x) = \omega x + b,$$

donde ω y b son los parámetros a encontrar. En los modelos de máquina de soporte vectorial se desea ajustar una banda de ancho ϵ que capture la mayor cantidad de elementos. En la Figura 3.1 se observa de color azul aquellos puntos que quedan fuera de la banda y que se consideran errores.

Para dimensiones mayores se desea hallar el hiperplano que mejor se ajuste, específicamente, si se considera a los puntos en un espacio $m+1$-dimensional $\bar{\omega} = (\omega_1, \omega_2, \ldots, \omega_m)$

y $\bar{x} = (x_1, x_2, \ldots, x_m)$, el hiperplano buscado es la función $\phi : \mathbb{R}^m \to \mathbb{R}$, dada por

$$\phi(\bar{x}) = \bar{\omega} \cdot \bar{x} + b.$$

Más general, se puede considerar a la ecuación como no lineal si se define como

$$\phi(\bar{x}) = \sum_{i=1}^{m} \omega_i K(\bar{x}, \bar{x}_i') + b,$$

donde a la función $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ se le conoce como *kernel* y a la familia de vectores $\{\bar{x}_i'\}_{i=1}^m$ se les denomina *vectores soporte*.

Existen diversas funciones de kernel, por mencionar algunas se tienen:

- Kernel lineal:

$$K(\bar{x}, \bar{x}') = \bar{x} \cdot \bar{x}'.$$

- Kernel polinomial de grado d entero distinto de cero:

$$K(\bar{x}, \bar{x}') = (\bar{x} \cdot \bar{x}' + 1)^d.$$

- Kernel gaussiano:

$$K(\bar{x}, \bar{x}') = e^{-\frac{\|\bar{x} - \bar{x}'\|^2}{2\sigma^2}}.$$

- Kernel sigmoide:

$$K(\bar{x}, \bar{x}') = \tanh(\beta \bar{x} \cdot \bar{x}' + b).$$

Para medir el ancho del margen de la banda en estos modelos se utilizan alguna de las métricas l_p. En los problemas de clasificación se utiliza la métrica euclidiana con $p = 2$:

$$\|\bar{x}\|_2 = \sqrt{\sum_{i=1}^{m} x_i^2}.$$

En los problemas de regresión se utiliza la norma l_1:

$$\|\bar{x}\|_1 = \sum_{i=1}^{m} |x_i|.$$

Ahora, el problema se formula como

$$\text{Minimizar } \frac{1}{m}\|\bar{\omega}\|_1 + C\sum_{i=1}^{m}\xi_i; \tag{3.2.1}$$

$$\text{sujeto a } \xi_i = \text{máx}\{\epsilon - y_i\phi(\bar{x}_i), 0\}. \tag{3.2.2}$$

El término $\frac{1}{m}\|\bar{\omega}\|_1$ se refiere a la determinación del vector que minimice el ancho de la banda, el término $\sum_{i=1}^{m}\xi_i$ es el error cometido y C es una constante de control del error que se establece a priori. La condición $\xi_i = \text{máx}\{1 - y_i\phi(\bar{x}_i), 0\}$ hace referencia a que el error está definido mediante la multiplicación del valor real conocido y_i y la predicha por el modelo $\phi(\bar{x}_i)$, en clasificación binaria, donde los valores solo toman 1 y -1, se toma $\epsilon = 1$ y si el modelo acierta el error es cero, si falla es dos.

Ahora, el problema (3.2.1)-(3.2.2) es equivalente al problema de optimización:

$$\min_{\bar{\omega},\xi,b}\left[\frac{1}{m}\sum_{i=1}^{m}|\omega_i| + C\sum_{i=1}^{m}\phi(\bar{x}_i)\right], \tag{3.2.3}$$

$$\text{sujeto a } y_i\phi(\bar{x}_i) \geq \epsilon - \xi_i. \tag{3.2.4}$$

Resuelto el problema (3.2.3)-(3.2.4), la función que representa al hiperplano es

$$\psi(\bar{x}) = \sum_{i=1}^{m}\omega_i^* K(\bar{x}, \bar{x}_i) + b^*, \tag{3.2.5}$$

donde $\omega^* = (\omega_1^*, \omega_2^*, \ldots, \omega_m^*)$ y b^* son los valores óptimos de los parámetros.

3.3. Maquina de soporte vectorial para selección de variables

Los modelos de máquina de soporte vectorial pueden ser adaptados para realizar una selección de variables. La idea es que las componentes del vector $\bar{\omega}$ reflejan la relevancia de cada variable, es decir, la magnitud y signo de la componente ω_i indica el efecto de la i-ésima variable en el modelo. Si $\omega_i > 0$, la variable contribuye a y; si $\omega_i < 0$, la variable reduce el valor de y. En esta sección se detallará la manera en la que se adapta esta idea para determinar aquellas variables con mayor influencia en la función (3.2.5).

Considérese el problema de regresión (3.2.3)-(3.2.4) como encontrar una función $f^* \in F = \{f : \mathbb{R}^n \to \mathbb{R}\}$ que minimiza el funcional

$$R[f] := P[f] + C\frac{1}{m}\sum_{i=1}^{m} L(y_i, f(\bar{x}_i).$$

La función $L(\cdot)$ se conoce como *función de pérdida*. De acuerdo con lo expuesto en la sección anterior, para los problemas de regresión esta función depende de ϵ, en específico, $L_\epsilon(y, f(x)) = \text{máx}\{|y - f(x)| - \epsilon, 0\}$. Luego, $P[\cdot]$ es un operador de regularización y C es llamado el parámetro de regularización. Para funciones lineales $f(\bar{x}) = \omega \cdot \bar{x} + b$, el operador de regularización es la norma l_2. En el caso de funciones no lineales, estas son producidas por medio del mapeo $\bar{x} \mapsto \Phi(\bar{x})$ vía el kernel K y se construyen funciones lineales dentro del espacio de llegada, que comúnmente se le llama *espacio de características*, de hecho, una función lineal en el espacio de características corresponde a una función no lineal en el espacio de entrada original ([5]).

Ahora, la solución óptima $\bar{\omega}$ del modelo de máquina de soporte vectorial puede ser expresado como

$$\bar{\omega} = \sum_{i=1}^{m} \alpha_i \Phi(\bar{x}_i).$$

Por lo tanto, la función de regresión puede ser expresada de forma equivalente como una expansión del kernel

$$f(\bar{x}) = \sum_{i=1}^{n} \alpha_i \Phi(\bar{x}_i) \cdot \Phi(\bar{x}) + b = \sum_{i=1}^{n} \alpha_i k(\bar{x}_i, \bar{x}) + b.$$

Los modelos clásicos de máquinas de soporte vectorial son programas cuadráticos en términos del vector $\bar{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ y resolverlos es más costoso computacionalmente, sin embargo, la estrategia que se utiliza es aplicar la norma l_1 al vector α en el término que contiene la expansión del kernel. Por lo que el funcional regularizado queda determinado por

$$R[f] := \sum_{i=1}^{m} |\alpha_i| + C\frac{1}{m}\sum_{i=1}^{m} L_\epsilon(y_i, f(x_i)),$$

recomendamos la referencia [5] donde se muestra a detalle el proceso.

De esta manera, se obtienen dos hiperparámetros: C y ϵ. El parámetro ϵ es el que define el tamaño del margen y, por lo regular, no es sencillo de determinar ya que se desconoce que tan exacta la función se ajustará. En este sentido, Schölkopf et al. en [11] y Smola et al. en [13], desarrollaron un ajuste automático utilizando un parámetro $\nu \in (0, 1]$, este parámetro proporciona una cota superior al error y una cota inferior a la fracción de vectores soporte. Para formar el modelo de selección de variables se reescribe $\alpha_i = u_i - v_i$, donde $u_i,\ v_i \geq 0$. La solución tiene o u_i o v_i igual a cero dependiendo del signo de α_i, por lo que $|\alpha_i| = u_i + v_i$.

Ahora, considérese la base de datos a reducir como $(\bar{x}_i, y_i)$, $i = 1, 2, \ldots, m$, con $\bar{x} \in \mathbb{R}^m$. La j-ésima componente de $\bar{x}_i$ es denotada por x_{ij}, $j = 1, 2, \ldots, n$. El problema de programación lineal a resolver es

$$\min_{\bar{u},\bar{v},b,\epsilon,\eta} \sum_{i=1}^{m}(u_i + v_i) + C\frac{1}{m}\sum_{i=1}^{m}(\xi_i + \eta_i) + C\nu\epsilon \tag{3.3.1}$$

$$\text{tal que } y_i - \sum_{i=1}^{m}(u_i - v_i)K(\bar{x}_i, \bar{x}_j) - b \leq \epsilon + \xi_i, \qquad i = 1, \ldots, l, \tag{3.3.2}$$

$$\sum_{i=1}^{m}(u_i - v_i)K(\bar{x}_i, \bar{x}_j) + b - y_i \leq \epsilon + \eta_i, \qquad i = 1, \ldots, m, \tag{3.3.3}$$

$$u_i,\ v_i,\ \xi_i,\ \eta_i\ \epsilon \geq 0, \qquad i, j = 1, \ldots, m. \tag{3.3.4}$$

El vector de parámetros η realiza la función de controlar el error, ya que como se observa en (3.3.1) aparece el término adicional $C\nu\epsilon$.

En la selección de variables realizada en esta investigación se utilizó un kernel lineal, esto es, se reemplaza $K(\bar{x}_i, \bar{x}_j)$ por x_{ij} donde el índice i corre sobre los registros y j sobre las variables.

Luego de resuelto el problema de programación lineal (3.3.1)-(3.3.4), la solución óptima es $\bar{\omega} = \bar{\alpha} = \bar{u} - \bar{v}$. Finalmente, para realizar la selección de variables, se establece un umbral γ y se eligen los valores ω_i tales que $\gamma \leq |\omega_i|$, $i = 1, \ldots, m$, luego se seleccionan las variables correspondientes a cada ω_i seleccionado.

En esta investigación, los valores utilizados son $C = 0.01$, $\epsilon = 0.0001$ y el valor umbral

es $\gamma = 0.00001$.

El resultado de la selección de variables son 31 de tipo API, 2 de Commands Signature, 3 de Intent y 18 de Manifest Permission, en total se reduce de 216 a 54 variables. En específico, las variables significativas son

- API:

    1. transact
    2. onServiceConnected
    3. Ljava.lang.Class.getCanonicalName
    4. Ljava.net.URLDecoder
    5. android.telephony.SmsManager
    6. getBinder
    7. ClassLoader
    8. Landroid.content.Context.registerReceiver
    9. Landroid.content.Context.unregisterReceiver
    10. getCallingUid
    11. Ljavax.crypto.spec.SecretKeySpec
    12. android.content.pm.PackageInfo
    13. TelephonyManager.getLine1Number
    14. HttpGet.init
    15. System.loadLibrary
    16. android.intent.action.SEND
    17. android.telephony.gsm.SmsManager
    18. TelephonyManager.getSubscriberId
    19. Ljava.lang.Object.getClass

20. Binder
21. TelephonyManager.getDeviceId
22. Ljava.lang.Class.getPackage
23. sendMultipartTextMessage
24. HttpUriRequest
25. Runtime.exec
26. TelephonyManager.getSimOperator
27. onBind
28. Ljava.lang.Class.getResource
29. findClass
30. TelephonyManager.getSimCountryIso_0
31. TelephonyManager.getSimCountryIso_1

- Commands signature:

32. chmod
33. /system/bin

- Intent:

34. android.intent.action.BOOT_COMPLETED
35. android.intent.action.PACKAGE_REPLACED
36. android.intent.action.TIME_SET

- Manifest Permission:

37. SEND_SMS
38. READ_PHONE_STATE
39. GET_ACCOUNTS

40. READ_SMS
41. USE_CREDENTIALS
42. MANAGE_ACCOUNTS
43. READ_SYNC_SETTINGS
44. WRITE_HISTORY_BOOKMARKS
45. CAMERA
46. READ_HISTORY_BOOKMARKS
47. INTERNET
48. ACCESS_LOCATION_EXTRA_COMMANDS
49. MODIFY_AUDIO_SETTINGS
50. ACCESS_COARSE_LOCATION
51. WRITE_SETTINGS
52. CHANGE_NETWORK_STATE
53. CALL_PHONE
54. WRITE_CONTACTS

Notesé que la mayor parte de las variables seleccionadas son del tipo API, lo cual se puede interpretar que el dispositivo corre más riesgo de ser infectado con un Malware por medio de los permisos otorgados a las aplicaciones que utilizan la interfaz de programación, es decir a aplicaciones externas de Android.

3.4. Maquina de soporte vectorial para clasificación

En el caso de clasificación binaria, la idea es hallar un hiperplano que separe a las dos clases en los que se encuentra la nube de puntos con el menor número de errores. Con la variable objetivo codificada como $+1$ y -1, y los parámetros óptimos obtenidos del

modelo de programación lineal (3.2.3)-(3.2.4), el hiperplano tiene como ecuación

$$\sum_{i=1}^{m} \omega_i^* K(\bar{x}, \bar{x}_i) + b^* = 0, \tag{3.4.1}$$

con $\bar{x}_i$ los vectores soporte.

Una parte importante de la clasificación, después de encontrar el hiperplano de separación, es la aplicación de una *función de decisión* para determinar la etiqueta de un nuevo registro. Para este estudio, la función de decisión se define como:

$$f(x) = \text{sgn} \left(\sum_{i=1}^{m} \omega_i^* K(\bar{x}, \bar{x}_i) + b^* \right),$$

con la cual se obtiene la clase de cada nuevo registro bajo la función f.

En este caso se hace la suposición que la colección de puntos son separables por hiperplanos, por ello se dice que son *problemas linealmente separables*. Para el caso en el que no sean de esta clase se demuestra que para dimensiones mayores sí lo son, luego se reduce la dimensión a la original, de esta manera se determina el hiperplano de separación. Recomendamos la lectura de la referencia [11] para adentrarse en este tema. Para el caso de la presente investigación, se mantendrá la suposición de la separación lineal. Con lo que el problema se reduce a hallar el hiperplano de separación a partir del conjunto de entrenamiento. Esta etapa será descrita a detalle en la siguiente subsección.

3.4.1. Entrenamiento

Entre todos los hiperplanos que separan los datos, se demuestra que existe un único hiperplano óptimo (véase [11]), que se caracteriza por tener el margen máximo de separación entre cualquier punto del conjunto de entrenamiento $\{\bar{x}_j\}_{j=1}^{k}$ y el hiperplano. Este es la solución de

$$\begin{aligned} \text{Maximizar}_{\ \omega \in \mathbb{R}^n, b \in \mathbb{R}} \ & \text{mín}\{\|\bar{x} - \bar{x}_j\|, \text{ tal que } \bar{x} \in \mathbb{R}^n, \\ & \sum_{i=1}^{m} \omega_i^* K(\bar{x}, \bar{x}_i) + b^* = 0, \ j = 1, \ldots, k\}. \end{aligned} \tag{3.4.2}$$

En la Sección 1.3 del libro [11] se establece que la clase de hiperplanos de separación decrece cuando el margen crece, con lo que se sustenta la existencia del hiperplano óptimo. Para construirlo, se debe resolver

$$\min_{\bar{\omega}\in\mathbb{R}^n, b\in\mathbb{R}} \tau(\omega) = \frac{1}{2}\|\omega\|^2 \tag{3.4.3}$$

$$\text{sujeto a } \; y_i \left(\sum_{i=1}^{m} \omega_i^* K(\bar{x}_j, \bar{x}_i) + b^* \right) \geq 1, \text{ para todo } i = 1, \ldots, k. \tag{3.4.4}$$

Las restricciones (3.4.4) garantizan que $f(\bar{x}_j)$ sea $+1$ para $y_i = +1$ y -1 para $y_i = -1$. También al pedir que la restricción sea mayor que 1 no se lograría minimizar τ (véase [11, Sección 1.4]).

La función τ en la ecuación (3.4.3) recibe el nombre de *Función Objetivo*, mientras que las ecuaciones (3.4.4) se llama *restricciones*, estas dos ecuaciones forman el llamado *problema de optimización restringida*, los cuales se solucionan introduciendo multiplicadores de *Lagrange* $\alpha_i \geq 0$ y un *Lagrangiano* L de la forma

$$L(\bar{\omega}, b, \bar{\alpha}) = \frac{1}{2}\|\omega\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \left(\sum_{i=1}^{m} \omega_i^* K(\bar{x}_j, \bar{x}_i) + b^* \right) - 1 \right). \tag{3.4.5}$$

Este lagrangiano se minimiza respecto a las variables ω y b, además de maximizarse con respecto a las variables duales α_i, es decir, se encontrará un punto silla (véase [11]). Por lo que las derivadas parciales de L con respecto a b y ω deben ser cero, esto es,

$$\frac{\partial}{\partial b} L(\bar{\omega}, b, \bar{\alpha}) = 0, \tag{3.4.6}$$

$$\frac{\partial}{\partial \omega} L(\bar{\omega}, b, \bar{\alpha}) = 0, \tag{3.4.7}$$

con lo que se obtiene

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.4.8}$$

y

$$\bar{\omega} = \sum_{i=1}^{m} \alpha_i y_i x_i. \tag{3.4.9}$$

Por lo tanto, el vector solución tiene la forma (3.4.9) en términos de los elementos del conjunto de entrenamiento. A los elementos $\alpha_i \neq 0$ se le denominan vectores soporte y cumplen

$$\alpha_i \left(y_i \left(\sum_{i=1}^{m} \omega_i^* K(\bar{x}_j, \bar{x}_i) + b^* \right) - 1 \right) = 0, \tag{3.4.10}$$

para todo $i = 1, ..., m$.

Al sustituir (3.4.8) y (3.4.9) en el Lagrangiano (3.4.5) se eliminan las variables $\bar{\omega}$ y b, con lo que se obtiene el llamado *problema de optimización dual*:

$$\text{Maximizar}_{\alpha \in \mathbb{R}^m} \ \Omega(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j, \tag{3.4.11}$$

$$\text{sujeto a} \ \ \alpha_i \geq 0, \ \text{para toda } i = 1, \cdots, m$$

$$\text{y} \ \sum_{i=1}^{m} \alpha_i y_i = 0.$$

Al resolver el problema dual se obtienen los coeficientes α, luego, al sustituir la igualdad (3.4.9) en la función de decisión, se tiene que

$$f(\bar{x}) = \text{sgn} \ \left(\sum_{i=1}^{m} y_i \alpha_i \bar{x} \cdot \bar{x}_i + b \right),$$

donde b se calcula de la igualdad (3.4.10). De esta manera, el modelo se encuentra entrenado y está listo para ser sometido a la validación de su desempeño y a posteriores simulaciones, las cuales se realizan en las subsecciones siguientes.

3.4.2. Validación

Una vez entrenado el modelo, se somete a un subconjunto base que se utiliza para las pruebas. Existen cuatro resultados posibles de lo que el modelo de clasificación podría hacer en cualquier instancia dada, estos son:

- **True positive (TP).** El clasificador etiqueta un elemento positivo como positivo, lo que resulta en una victoria para el clasificador.
- **False positive (FP).** El clasificador erróneamente llama positivo a un elemento negativo, lo que resulta en un error de clasificación tipo I.

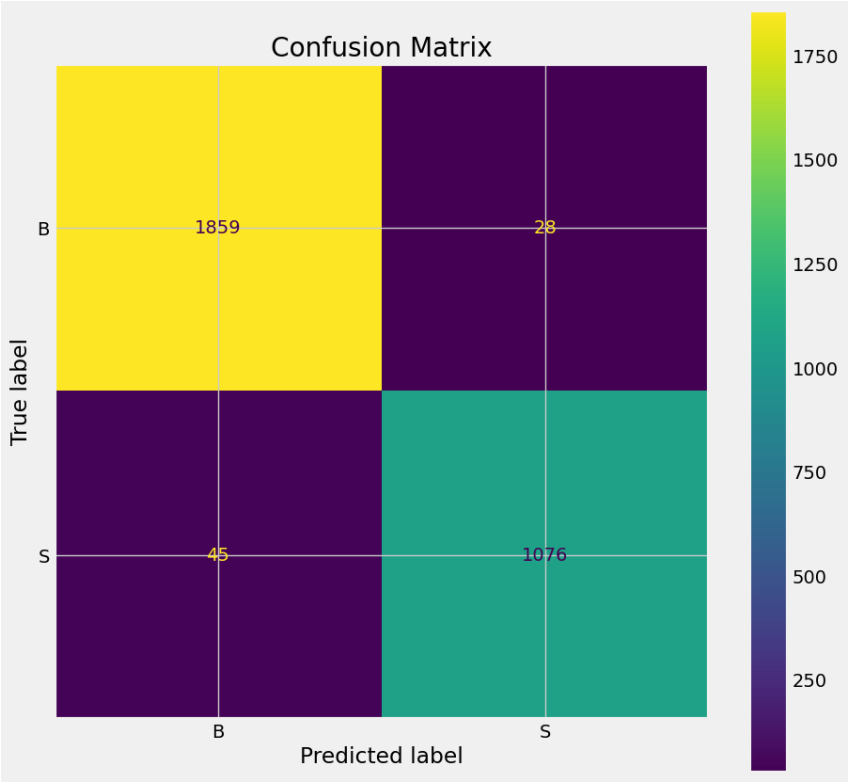

Figura 3.2: Matriz de confusión del entrenamiento.

- **False negative (FN).** El clasificador declara erróneamente un elemento positivo como negativo, lo que resulta en un error de clasificación tipo II.

- **True Negative (TN).** Aquí el clasificador determina correctamente que un miembro de la clase negativa merece una etiqueta negativa.

Para el modelo, los resultados son desplegados en la Figura 3.2, donde se observa un cuadro que recibe el nombre de *matriz de confusión*.

Con los resultados del número de aciertos y fallos ilustrados en la matriz de confusión se calculan las métricas de desempeño. Estas métricas miden el desempeño del modelo y se determina la confiabilidad en la clasificación, de aquí el nombre de *validación del modelo*.

Las métricas dependen del tipo de modelo que se utiliza, para modelos clasificadores binarios, como lo es el presente caso, se tienen cuatro métricas:

- **Accuracy.** Es la precisión del clasificador y se calcula mediante

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP}. \tag{3.4.12}$$

- **Precision.** La precisión mide la frecuencia con la que este clasificador es correcto, la manera de calcularlo es

$$precision = \frac{TP}{(TP + FP)}. \tag{3.4.13}$$

- **Recall.** El recall mide la frecuencia con la que tienes razón en todos los casos positivos:

$$recall = \frac{TP}{(TP + FN)}. \tag{3.4.14}$$

  Un alto recall implica que el clasificador tiene pocos falsos negativos.

- **F1 score.** También llamado *F-score*. Esta métrica es la que arroja menos ambigüedad, debido a que es una combinación de presicion y de recall que devuelve la media armónica entre ellos:

$$F = \frac{precision * recall}{precision + recall}. \tag{3.4.15}$$

  Un valor alto del F1-score requiere un recall y un precision alto.

Los valores obtenidos de la matriz de confusión son TP= 1859, TN= 1076, FP= 45 y FN= 28; utilizando estos valores se obtienen las estadísticas del Cuadro 3.4.1.

Estadística	Puntuación
Accuracy	97.57313829787235
Precision	97.63655462184873
Recall	98.51616322204558
F1 Score	98.07438670535478

Cuadro 3.4.1: Métricas de desempeño del modelo clasificatorio.

A continuación, se comparará el desempeño del modelo con algunos existentes. En el articulo *DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware*

Detection ([17]) se tienen algunas métricas de diversos modelos; dichos modelos son el resultado de combinar algoritmos basados en clasificación, estos son:

- *AAB:* Clasificación basada en la precisión promedio.
- *CDB:* Clasificación basada en el diferencial de clases.
- *RAPC:* Clasificación basada en el desempeño por clase.
- *RACD:* Clasificación de precisión promedio basado en el diferencial de clases.

La combinación de estos algoritmos lo realizan con el objetivo de obtener mejores métricas que empleando clasificadores ya definidos. En el Cuadro 3.4.2 se pueden observar estas métricas en comparación con las obtenidas en este trabajo, en precisión el valor se encuentra por encima de la media, además, lo más destacable es que en la métrica recall lo obtenido supera a todas las combinaciones realizadas en el artículo, esto significa que SVM tiene menos falsos positivos, lo que lo convierte en un modelo más confiable.

Modelos	**Precision**	**Recall**
AAB+CDB	96.6	97.2
AAB+RAPC	98.4	97.2
AAB+RACD	96.6	97.2
CDB+RAPC	98.1	97.2
CDB+RACD	96.6	97.2
RAPC+RACD	98.1	97.6
SVM	97.63655462	98.51616322

Cuadro 3.4.2: Comparación de las métricas de desempeño de los modelos. Los primeros seis corresponden a los establecidos en [17]. El último es la propuesta de esta investigación.

3.4.3. Simulación

La base de datos empleada en el proceso de simulación esta conformada por las 54 variables que resultaron del proceso de selección de variables, algunos de los valores del

registro que se introducen se muestran en la Tabla 3.4.3.

transact	1
onServiceConnected	0
SEND_SMS	1
Ljava.lang.Class.getCanonicalName	0
Ljava.net.URLDecoder	0
android.telephony.SmsManager	1
READ_PHONE_STATE	1
getBinder	0
ClassLoader	0
Landroid.content.Context.registerReceiver	1
Landroid.content.Context.unregisterReceiver	1
GET_ACCOUNTS	0

Cuadro 3.4.3: Cuadro con algunas de las variables y valor asignado al registro sometido en la simulación.

Dichas variables solo tienen asignado un único valor, ya que representan a la información obtenida del dispositivo móvil en cuestión.

Como resultado de la introducción del registro en el modelo entrenado para la predicción (ver Anexo 6), se obtiene el valor -1, por lo que se interpreta como Falso, es decir, el dispositivo no tiene Malware en las aplicaciones que lo conforman.

Conclusiones

El objetivo general de la investigación fue proporcionar un programa de aprendizaje máquina predictivo que muestre riesgos de ataques de malware a sistemas Android. Para lograrlo, en el Capítulo 1 se muestra el marco conceptual relacionado con las variables, debido a que se observó que era necesario dedicar un espacio para el reconocimiento del problema y de los conceptos básicos para explicar la problemática. Luego, se muestra las principales clases en las que se dividen las variables: API call signature, Commands signature, Intent y Manifest Permission, con sus respectivas descripciones y se muestran algunas de las variables que pertenecen a cada clase.

El Capítulo 2 está dedicado al análisis exploratorio de la base, se inicia con la abstracción de la base en forma matemática, se identifica como una base de datos mixta (o mezclada), se observa que las variables son binarias, lo que se espera en un tema relacionado con tecnología y cómputo. Parte importante que se descubre es que no cuenta con datos perdidos o faltantes, lo que permite que la base tenga una excelente usabilidad. En el mismo capítulo se propone un problema de clasificación para solucionar este problema, ya que la variable objetivo es binaria: con riesgo de invasión y sin riesgo de invasión. Además, se describe brevemente las fases de entrenamiento y de prueba del algoritmo a utilizar. A continuación, se realiza la representación de las variables en forma algebraica para realizar el proceso y se muestran algunos diagramas de barras donde se observa la distribución de frecuencias de ellas, se muestra también para la variable objetivo denominada *"class"*

y se compone de dos tipos: B y S, que representan las palabras "seguro" y "susceptible". Posterior a ello, se realiza un análisis de correlación y se observa una alta correlación entre todas las variables, sin embargo, se decide realizar un estudio de selección de variables para elegir a las que tienen mayor efecto en el fenómeno.

En el Capítulo 3 se plantea el modelo a utilizar: *Maquina de soporte vectorial*, sin embargo, la utilidad de este tipo de modelos es tan amplia que se decide utilizarlo tanto para la selección de variables como para la clasificación, por esta razón, este capítulo inicia con la descripción de los problemas de clasificación y regresión, luego, en la Sección 3.2 se da una descripción del funcionamiento y de la idea detrás de esta clase de modelos para contextualizar al algoritmo, esto da paso a la selección de variables planteada en la Sección 3.3, en donde se muestra la formulación matemática del modelo de regresión para que se logre seleccionar las variables relevantes. Con este proceso se observa que algunas de las variables significativas son: *transact*, *onServiceConnected*, *getBinder*, *chmod*, */system/bin*, *android.intent.action.BOOT_COMPLETED*, *SEND_SMS*, *READ_PHONE_STATE*, *GET_ACCOUNTS*, *READ_SMS* y *USE_CREDENTIALS*, de estas variables 31 son API, 2 son Commands Signature, 3 son Intent y 18 son Manifest Permission.

Con lo anterior realizado, en la Sección 3.4 se establece a detalle la forma en la que se realiza la clasificación mediante la obtención del hiperplano de separación y de la aplicación de la función de decisión. Esta sección contiene tres subsecciones, en la primera se muestra el funcionamiento de la fase de entrenamiento, en la segunda la de validación y en la tercera la de simulación. Derivado de esto, se determina que el modelo tiene un desempeño muy bueno, ya que comparado con los existentes en la literatura se encuentra dentro del promedio con:

- Accuracy : 97.57313829787235
- Precision : 97.63655462184873
- Recall : 98.51616322204558

- F1 Score : 98.07438670535478

Para finalizar, el modelo presentado tiene un excelente desempeño en la predicción, por lo que es posible utilizarlo para el monitoreo del riesgo potencial de un ataque de malware a los dispositivos con sistema operativo Android, solo verificando los valores de las variables significativas obtenidas en esta investigación, lo que podría acelerar la vigilancia. Con todo lo anterior, se logra el objetivo general y se comprueban las hipótesis originalmente planteadas.

Bibliografía

[1] R. Adeva. Qué es android: todo sobre el sistema operativo de google. *https://www.adslzone.net/reportajes/software/que-es-android/*, 3 de marzo de 2021.

[2] C. C. Aggarwal. *Data Mining: The Textbook*. Springer, 2015.

[3] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck. Drebin: effective and explainable detection of Android malware in your pocket. In *Ndss*, pages 23–26. 2014.

[4] Y. Bai, Z. Sun, B. Zeng, Long J., L. Li, J. Valente de Oliveira, and C. Li. A comparison of dimension reduction techniques for support vector machine modeling of multi-parameter manufacturing quality predictio. *Journal of Intelligent Manufacturing*, 2018.

[5] Jinbo Bi, Kristin P. Bennett, Mark J. Embrechts, Curt M. Breneman, and Minghu Song. Dimensionality reduction via sparse support vector machines. *J. Mach. Learn. Res.*, 3:1229–1243, 2003.

[6] Concepto. Malware. *https://concepto.de/malware/*.

[7] C. Cortes and V. Vapnik. *Machine Learning*, chapter Support-vector networks. 20(3), 1995.

[8] iProUP. La historia de android, el sistema operativo móvil que transformó el mundo. *https://www.iproup.com/coffee-break/16758-la-historia-de-android-el-sistema-operativo-que-cambio-el-mundo*, 14 de septiembre de 2020.

[9] McAfee Labs. Mcafee labs threat predictions report. March 2016.

[10] Malwarebytes. Malware. *https://es.malwarebytes.com/malware/*.

[11] B. Scholkopf, A.J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, 2000 May.

[12] S. S. Skiena. *The Data Science Design Manual*. Springer, 2017.

[13] A. Smola, B. Scholkopf, and G. Ratsch. Linear programs for automatic accuracy control in regression. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 575–580 vol.2, 1999.

[14] Statista. Smartphone os market share worldwide 2009-2015 statistic. Accedido el 10-01-2024 a `https://www.statista.com/statistics/263453/global-market-`, 2017.

[15] J. W. Tukey. The future of data analysis. *The Annals of Mathematical Statistics*, 33(1):1–67.

[16] S. Y. Yerima and S. Sezer. Droidfusion: A novel multilevel classifier fusion approach for android malware detection. *IEEE Transactions on Cybernetics*, pages 1—-14, 2018.

[17] S. Y. Yerima and S. Sezer. Droidfusion: A novel multilevel classifier fusion approach for android malware detection. *IEEE Transactions on Cybernetics*, 49(2):453–466, 2019.

[18] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *IEEE Symp. Secur. Priv.*, number 4, pages 95–109, 2012.

Anexo 1

#	**NOMBRE**	**TIPO**
1	transact	API call signature
2	onServiceConnected	API call signature
3	bindService	API call signature
4	attachInterface	API call signature
5	ServiceConnection	API call signature
6	android.os.Binder	API call signature
7	Ljava.lang.Class.getCanonicalName	API call signature
8	Ljava.lang.Class.getMethods	API call signature
9	Ljava.lang.Class.cast	API call signature
10	Ljava.net.URLDecoder	API call signature
11	android.content.pm.Signature	API call signature
12	android.telephony.SmsManager	API call signature
13	getBinder	API call signature
14	ClassLoader	API call signature
15	Landroid.content.Context.registerReceiver	API call signature
16	Ljava.lang.Class.getField	API call signature
17	Landroid.content.Context.unregisterReceiver	API call signature

#	**NOMBRE**	**TIPO**
18	Ljava.lang.Class.getDeclaredField	API call signature
19	getCallingUid	API call signature
20	Ljavax.crypto.spec.SecretKeySpec	API call signature
21	android.content.pm.PackageInfo	API call signature
22	KeySpec	API call signature
23	TelephonyManager.getLine1Number	API call signature
24	DexClassLoader	API call signature
25	HttpGet.init	API call signature
26	SecretKey	API call signature
27	Ljava.lang.Class.getMethod	API call signature
28	System.loadLibrary	API call signature
29	android.intent.action.SEND	API call signature
30	Ljavax.crypto.Cipher	API call signature
31	android.telephony.gsm.SmsManager	API call signature
32	TelephonyManager.getSubscriberId	API call signature
33	Runtime.getRuntime	API call signature
34	Ljava.lang.Object.getClass	API call signature
35	Ljava.lang.Class.forName	API call signature
36	Binder	API call signature
37	IBinder	API call signature
38	android.os.IBinder	API call signature
39	createSubprocess	API call signature
40	URLClassLoader	API call signature
41	abortBroadcast	API call signature
42	TelephonyManager.getDeviceId	API call signature
43	getCallingPid	API call signature
44	Ljava.lang.Class.getPackage	API call signature

#	**NOMBRE**	**TIPO**
45	Ljava.lang.Class.getDeclaredClasses	API call signature
46	PathClassLoader	API call signature
47	TelephonyManager.getSimSerialNum	API call signature
48	Runtime.load	API call signature
49	TelephonyManager.getCallState	API call signature
50	TelephonyManager.getSimCountryIso	API call signature
51	sendMultipartTextMessage	API call signature
52	PackageInstaller	API call signature
53	sendDataMessage	API call signature
54	HttpPost.init	API call signature
55	Ljava.lang.Class.getClasses	API call signature
56	TelephonyManager.isNetworkRoaming	API call signature
57	HttpUriRequest	API call signature
58	divideMessage	API call signature
59	Runtime.exec	API call signature
60	TelephonyManager.getNetworkOp	API call signature
61	MessengerService	API call signature
62	IRemoteService	API call signature
63	SET_ALARM	API call signature
64	ACCOUNT_MANAGER	API call signature
65	TelephonyManager.getSimOperator	API call signature
66	onBind	API call signature
67	Process.start	API call signature
68	Context.bindService	API call signature
69	ProcessBuilder	API call signature
70	Ljava.lang.Class.getResource	API call signature
71	defineClass	API call signature

#	**NOMBRE**	**TIPO**
72	findClass	API call signature
73	Runtime.loadLibrary	API call signature
74	mount	Commands sign.
75	chmod	Commands sign.
76	remount	Commands sign.
77	chown	Commands sign.
78	/system/bin	Commands sign.
79	/system/app	Commands sign.
80	android.intent.action.BOOT_COMPLETED	Intent
81	android.intent.action.PACKAGE_REPLACED	Intent
82	android.intent.action.SEND_MULTIPLE	Intent
83	android.intent.action.TIME_SET	Intent
84	android.intent.action.PACKAGE_REMOVED	Intent
85	android.intent.action.TIMEZONE_CHANGED	Intent
86	android.intent.action.ACTION_POWER_DISC	Intent
87	android.intent.action.PACKAGE_ADDED	Intent
88	android.intent.action.ACTION_SHUTDOWN	Intent
89	android.intent.action.PACKAGE_DATA_CLEAr	Intent
90	android.intent.action.PACKAGE_CHANGED	Intent
91	android.intent.action.NEW_OUTGOING_CALL	Intent
92	android.intent.action.SENDTO	Intent
93	android.intent.action.CALL	Intent
94	android.intent.action.SCREEN_ON	Intent
95	android.intent.action.BATTERY_OKAY	Intent
96	android.intent.action.PACKAGE_RESTARTED	Intent
97	android.intent.action.CALL_BUTTON	Intent
98	android.intent.action.SCREEN_OFF	Intent

#	**NOMBRE**	**TIPO**
99	intent.action.RUN	Intent
100	android.intent.action.SET_WALLPAPER	Intent
101	android.intent.action.BATTERY_LOW	Intent
102	android.intent.action.ACTION_POWER_CONN	Intent
103	SEND_SMS	Manifest Permission
104	READ_PHONE_STATE	Manifest Permission
105	GET_ACCOUNTS	Manifest Permission
106	RECEIVE_SMS	Manifest Permission
107	READ_SMS	Manifest Permission
108	USE_CREDENTIALS	Manifest Permission
109	MANAGE_ACCOUNTS	Manifest Permission
110	WRITE_SMS	Manifest Permission
111	READ_SYNC_SETTINGS	Manifest Permission
112	AUTHENTICATE_ACCOUNTS	Manifest Permission
113	WRITE_HISTORY_BOOKMARKS	Manifest Permission
114	INSTALL_PACKAGES	Manifest Permission
115	CAMERA	Manifest Permission
116	WRITE_SYNC_SETTINGS	Manifest Permission
117	READ_HISTORY_BOOKMARKS	Manifest Permission
118	INTERNET	Manifest Permission
119	RECORD_AUDIO	Manifest Permission
120	NFC	Manifest Permission
121	ACCESS_LOCATION_EXTRA_COMMANDS	Manifest Permission
122	WRITE_APN_SETTINGS	Manifest Permission
123	BIND_REMOTEVIEWS	Manifest Permission
124	READ_PROFILE	Manifest Permission
125	MODIFY_AUDIO_SETTINGS	Manifest Permission

#	**NOMBRE**	**TIPO**
126	READ_SYNC_STATS	Manifest Permission
127	BROADCAST_STICKY	Manifest Permission
128	WAKE_LOCK	Manifest Permission
129	RECEIVE_BOOT_COMPLETED	Manifest Permission
130	RESTART_PACKAGES	Manifest Permission
131	BLUETOOTH	Manifest Permission
132	READ_CALENDAR	Manifest Permission
133	READ_CALL_LOG	Manifest Permission
134	SUBSCRIBED_FEEDS_WRITE	Manifest Permission
135	READ_EXTERNAL_STORAGE	Manifest Permission
136	VIBRATE	Manifest Permission
137	ACCESS_NETWORK_STATE	Manifest Permission
138	SUBSCRIBED_FEEDS_READ	Manifest Permission
139	CHANGE_WIFI_MULTICAST_STATE	Manifest Permission
140	WRITE_CALENDAR	Manifest Permission
141	MASTER_CLEAR	Manifest Permission
142	UPDATE_DEVICE_STATS	Manifest Permission
143	WRITE_CALL_LOG	Manifest Permission
144	DELETE_PACKAGES	Manifest Permission
145	GET_TASKS	Manifest Permission
146	GLOBAL_SEARCH	Manifest Permission
147	DELETE_CACHE_FILES	Manifest Permission
148	WRITE_USER_DICTIONARY	Manifest Permission
149	REORDER_TASKS	Manifest Permission
150	WRITE_PROFILE	Manifest Permission
151	SET_WALLPAPER	Manifest Permission
152	BIND_INPUT_METHOD	Manifest Permission

#	**NOMBRE**	**TIPO**
153	READ_SOCIAL_STREAM	Manifest Permission
154	READ_USER_DICTIONARY	Manifest Permission
155	PROCESS_OUTGOING_CALLS	Manifest Permission
156	CALL_PRIVILEGED	Manifest Permission
157	BIND_WALLPAPER	Manifest Permission
158	RECEIVE_WAP_PUSH	Manifest Permission
159	DUMP	Manifest Permission
160	BATTERY_STATS	Manifest Permission
161	ACCESS_COARSE_LOCATION	Manifest Permission
162	SET_TIME	Manifest Permission
163	WRITE_SOCIAL_STREAM	Manifest Permission
164	WRITE_SETTINGS	Manifest Permission
165	REBOOT	Manifest Permission
166	BLUETOOTH_ADMIN	Manifest Permission
167	BIND_DEVICE_ADMIN	Manifest Permission
168	WRITE_GSERVICES	Manifest Permission
169	KILL_BACKGROUND_PROCESSES	Manifest Permission
170	STATUS_BAR	Manifest Permission
171	PERSISTENT_ACTIVITY	Manifest Permission
172	CHANGE_NETWORK_STATE	Manifest Permission
173	RECEIVE_MMS	Manifest Permission
174	SET_TIME_ZONE	Manifest Permission
175	CONTROL_LOCATION_UPDATES	Manifest Permission
176	BROADCAST_WAP_PUSH	Manifest Permission
177	BIND_ACCESSIBILITY_SERVICE	Manifest Permission
178	ADD_VOICEMAIL	Manifest Permission
179	CALL_PHONE	Manifest Permission

#	**NOMBRE**	**TIPO**
180	BIND_APPWIDGET	Manifest Permission
181	FLASHLIGHT	Manifest Permission
182	READ_LOGS	Manifest Permission
183	SET_PROCESS_LIMIT	Manifest Permission
184	MOUNT_UNMOUNT_FILESYSTEMS	Manifest Permission
185	BIND_TEXT_SERVICE	Manifest Permission
186	INSTALL_LOCATION_PROVIDER	Manifest Permission
187	SYSTEM_ALERT_WINDOW	Manifest Permission
188	MOUNT_FORMAT_FILESYSTEMS	Manifest Permission
189	CHANGE_CONFIGURATION	Manifest Permission
190	CLEAR_APP_USER_DATA	Manifest Permission
191	CHANGE_WIFI_STATE	Manifest Permission
192	READ_FRAME_BUFFER	Manifest Permission
193	ACCESS_SURFACE_FLINGER	Manifest Permission
194	BROADCAST_SMS	Manifest Permission
195	EXPAND_STATUS_BAR	Manifest Permission
196	INTERNAL_SYSTEM_WINDOW	Manifest Permission
197	SET_ACTIVITY_WATCHER	Manifest Permission
198	WRITE_CONTACTS	Manifest Permission
199	BIND_VPN_SERVICE	Manifest Permission
200	DISABLE_KEYGUARD	Manifest Permission
201	ACCESS_MOCK_LOCATION	Manifest Permission
202	GET_PACKAGE_SIZE	Manifest Permission
203	MODIFY_PHONE_STATE	Manifest Permission
204	CHANGE_COMPONENT_ENABLED_ST	Manifest Permission
205	CLEAR_APP_CACHE	Manifest Permission
206	SET_ORIENTATION	Manifest Permission

#	**NOMBRE**	**TIPO**
207	READ_CONTACTS	Manifest Permission
208	DEVICE_POWER	Manifest Permission
209	HARDWARE_TEST	Manifest Permission
210	ACCESS_WIFI_STATE	Manifest Permission
211	WRITE_EXTERNAL_STORAGE	Manifest Permission
212	ACCESS_FINE_LOCATION	Manifest Permission
213	SET_WALLPAPER_HINTS	Manifest Permission
214	SET_PREFERRED_APPLICATIONS	Manifest Permission
215	WRITE_SECURE_SETTINGS	Manifest Permission
216	class	B=Benign; S=Malware

Anexo 2

La indicación en python que da las correlaciones es:

```
datos.corr()
```

La instrucción para obtener el mapa de calor de las correlaciones es:

```
import seaborn as sns
plt.figure(figsize=(20,20))
sns.heatmap(data=round(datos.corr(),2), annot=True)
```

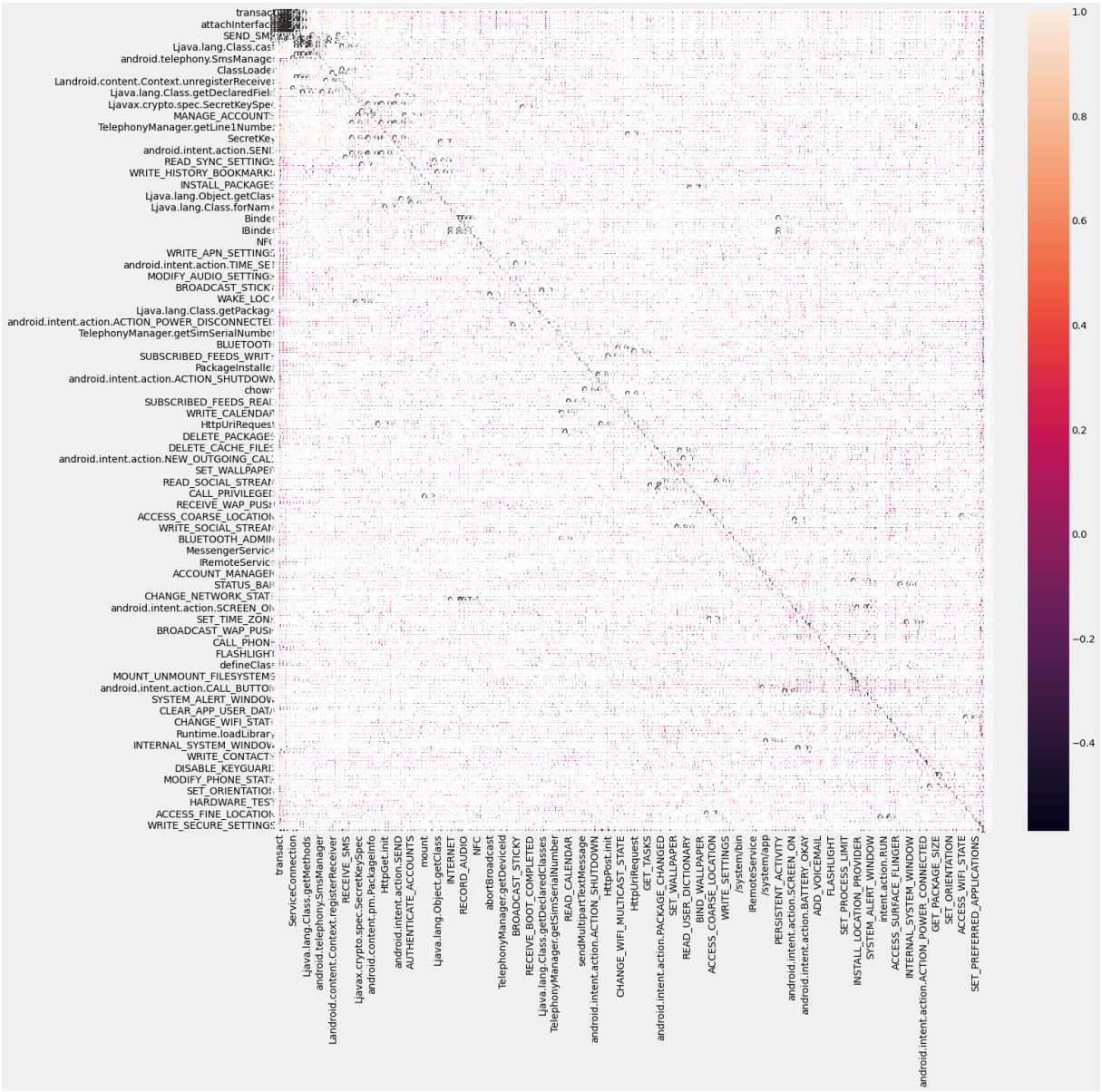

Figura 3.3: Gráfica de calor de la correlación de las variables.

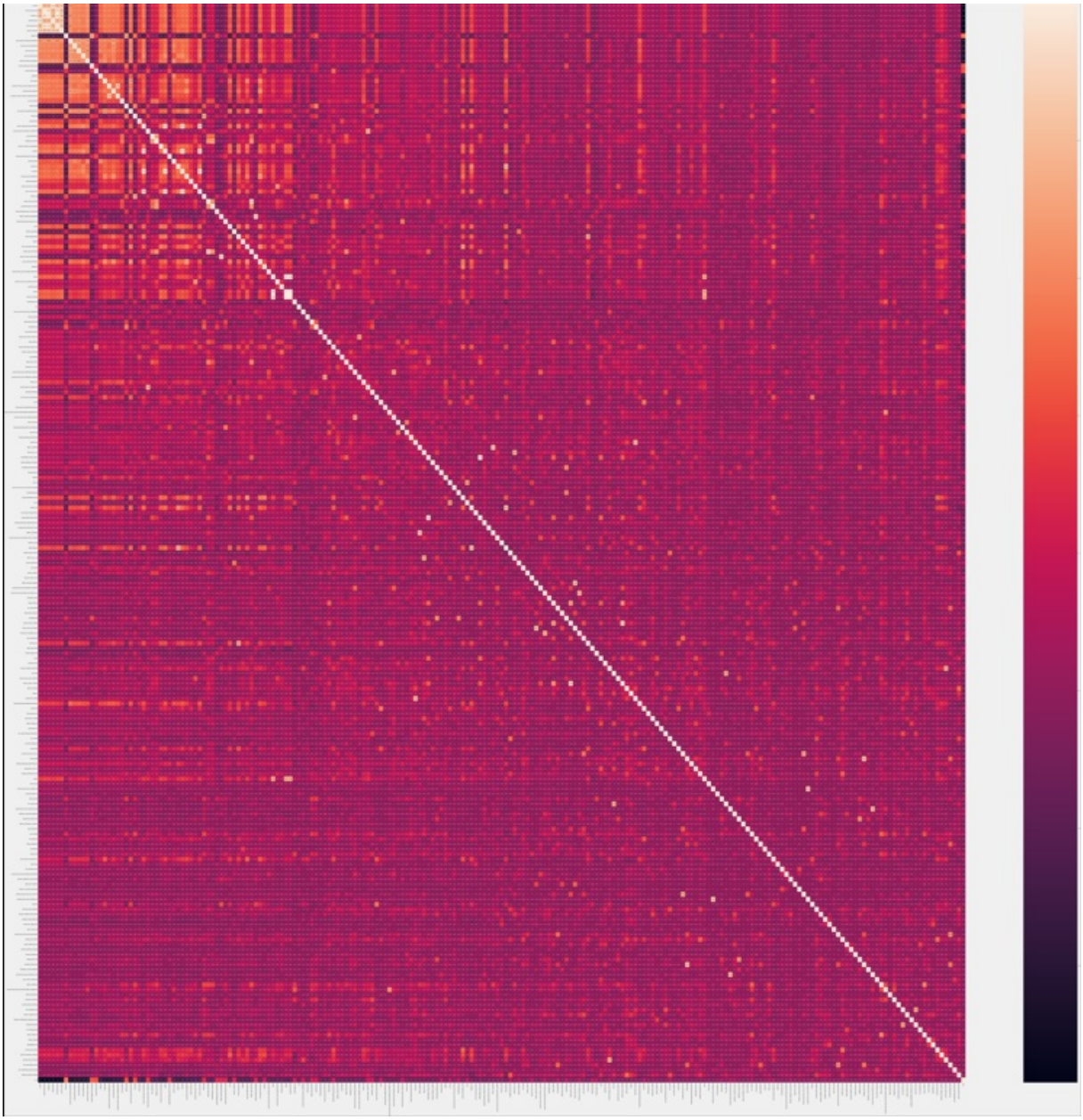

Figura 3.4: Tabla con todas las correlaciones de las variables.

Anexo 3

Se inicia con la exploración de la base de datos. Es procesada en un Notebook de *Colab*. Las líneas siguientes muestran la carga de las librerías necesarias:

```
import numpy as np # Algebra lineal
import pandas as pd # Manejo de datos
import matplotlib.pyplot as plt
```

A continuación, se carga la base de datos al entorno virtual:

```
from google.colab import files
uploads = files.upload()
```

Y se llama *datos*:

```
datos =
pd.read_csv('drebin-215-dataset-5560malware-9476-benign.csv')
```

Se carga **!pip instala basemap** y se importan las siguientes librerías:

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import seaborn as sns
import numpy as np
```

```
plt.style.use('fivethirtyeight')
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
from mpl_toolkits.basemap import Basemap
```

Estas librerías permiten graficar los datos; Para poder observar de una manera más eficaz si hacen falta algunos datos en la base, esto por medio de las siguientes instrucciones:

```
plt.figure(figsize=(16,6))
sns.heatmap(datos.isnull(),cmap='viridis',cbar=False)
```

La siguiente instrucción permiten ver los tipos de datos que están conformadas las variables, además de verificar cuantos nulos o faltantes existen en la base.

```
    datos.dtypes
    datos.isnull().sum()
```

Para saber cuantos datos están clasificados en los dos rubros se ejecutó el siguiente comando.

```
classes,count = np.unique(datos['class'],return_counts=True)
print(classes,count)
```

Con la finalidad de observar de manera gráfica cuantos datos son Malos o Buenos se utiliza lo siguiente.

```
print(classes,count)
plt.bar(classes,count)
plt.title("Equilibrio de clases")
plt.xlabel("Clases")
plt.ylabel("Contador")
plt.show()
```

Se imprime cada histograma de cada variable con las siguientes lineas.

```
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1
    and nunique[col] < 50]]
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None,
    figsize = (6 * nGraphPerRow, 8 * nGraphRow),
    dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]),
        np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```

```
plotPerColumnDistribution(datos, 215, 5)
```

Ahora se codifica la variable *class* que esta definida como "B" y "S", ahora serán "B=0" y "1=S".

```
label_encoder = preprocessing.LabelEncoder()
df_label= label_encoder.fit_transform(datos['class'])
datos['class']=df_label
datos['class']
```

Se calculan las correlaciones con las siguientes lineas.

```
# Calcular correlaciones
datos.corr()
```

Anexo 4

A fin de comenzar la extracción de características importamos la libreria sklearn.

```
from sklearn import preprocessing
```

Se crean los vectores de característica y objetivo, además de que se verifican que este conformada las columnas de constantes:

```
X = datos.drop("class", axis=1) #vector caracteristica
y = datos["class"] # vector objetivo.


#Caracteristicas constantes
[col for col in X.columns if X[col].nunique() == 1]
```

Las librerías a emplear para poder dividir matrices en subconjuntos de entrenamiento y prueba, además de codificar las etiquetas.

```
# Llamamos las librerias
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder


# Codificamos la variable objetivo
```

```
lr = LabelEncoder()
y = lr.fit_transform(y)
```

Se importan las librerias:

```
from sklearn.svm import LinearSVC
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectFromModel
```

get_dummies convierte variables categóricas en variables ficticias/indicadoras.

```
X= pd.get_dummies(X)
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
```

```
class MultiColumnLabelEncoder:
    def __init__(self,columns = None):
        self.columns = columns # array of column names to encode


    def fit(self,X,y=None):
        return self # not relevant here


    def transform(self,X):
        '''
        Transforms columns of X specified in self.columns using
        LabelEncoder(). If no columns specified, transforms all
        columns in X.
        '''
        output = X.copy()
        if self.columns is not None:
            for col in self.columns:
```

```
                output[col] = LabelEncoder().fit_transform(output
                [col])
        else:
            for colname,col in output.iteritems():
                output[colname] = LabelEncoder().fit_transform
                (col)
        return output

    def fit_transform(self,X,y=None):
        return self.fit(X,y).transform(X)
```

En el siguiente paso se hace la selección de características.

```
selector = SelectFromModel(estimator=LinearSVC(C=0.01,
penalty="l1", dual=False)).fit(X, y)
```

```
selector.estimator_
```

```
selector.n_features_in_
```

Para saber cuales son las variables con las que trabaja el modelo:

```
selector.feature_names_in_
```

Los coeficientes de estas variables son

```
selector.estimator_.coef_
```

Donde el umbral que resulta para seleccionar la variable se averigua con el código:

```
selector.threshold_
```

```
lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y)
```

```
model = SelectFromModel(lsvc, prefit=True)
```

Se crea una base de datos a partir de las variables resultantes de la selección.

```
cols = model.get_support(indices=True)
features_df_new_lsvc = X.iloc[:,cols]
```

Se corroboran lo tipos de datos y su estatus:

```
features_df_new_lsvc.dtypes
status = model.get_support()
print("Selection status: ", status)
```

Se carga la Liberaría numpy que permite manejar arreglos:

```
from numpy import array
features = array(datos.columns)
print("All features:")
print(features)
```

Obtención columnas para conservar y crear un nuevo marco de datos solo con esas:

```
cols = model.get_support(indices=True)
features_df_new_lsvc = X.iloc[:,cols]
```

Se corrobora la cantidad de variables generadas:

```
features_df_new_lsvc.head()
```

Anexo 5

Dado que ya se cargaron las librerías a utilizar en el el anexo anterior, se comienza directamente con la indicación del modelo para su entrenamiento:

```
X = features_df_new_lsvc
y = datos["class"]
```

```
train_x,test_x,train_y,test_y = train_test_split(X.to_numpy(),
        y.to_numpy(),
          test_size = 0.2,
            shuffle=True)
```

Se entrena:

```
lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y)
```

Se importan de nuevo las librerias:

```
import pandas as pd
import numpy as np
np.random.seed(0)
from sklearn.metrics import precision_score,recall_score,f1_score
import tensorflow as tf
tf.compat.v1.set_random_seed(0)
```

```
from tensorflow import keras
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
```

La función que realizará la función del predictor será:

```
y_pred = lsvc.predict(test_x)
```

Impresión de la matriz de confusión para verificar la calidad de el predictor:

```
classes = ["B","S"]
cm = confusion_matrix(y_pred,test_y)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=classes)
fig, ax = plt.subplots(figsize=(10,10))
plt.title("Confusion Matrix")
disp = disp.plot(ax=ax)
plt.show()
```

Anexo 6

Se carga la base de datos con la cual se va a predecir.

```
X = features_df_new_lsvc
y = datos["class"]
```

Instrucción para realizar la predicción:

```
mi_prediccion=lsvc.predict(datos_de_prueba_lsvc)
mi_prediccion
```