3

Ciencias y Tecnologías

INTRODUCCIÓN A LOS SISTEMAS COMPUTACIONALES. UNA APROXIMACIÓN DEL ESTADO DEL ARTE

AUTORES:

Ing. Alba Marisol Córdova Vaca, Mg.

Ing. Geovanny Euclides Silva Peñafiel, Mg.

Ing. Victor Alfonso Cusco Vinueza, Mg.

Ing. Jaime Mesias Cajas, Mg.

Ing. Doris Karina Chicaiza Angamarca, Mg.

Ing. Franklin Javier Montaluisa Yugla, Mg.

EDITORIAL ALEMA

ISBN: 978-9942-7134-7-6

Primera Edición 2023
ISBN: 978-9942-7134-7-6
2023, ALEMA Casa Editora-Editorial Internacional S.A.S.D
Calle Simón Bolívar. A 200 metros del Parque Central de Jipijapa. Jipijapa, Ecuador.
https://editorialalema.org/libros/index.php/alema

Diseño y diagramación:
Ing. Wilter Leonel Solórzano Álava, Mg.
Corrección de contenidos:
Dr. C. Omar Mar Cornelio, PhD.
Diseño, montaje y producción editorial:
ALEMA Casa Editora-Editorial Internacional S.A.S.D, Ecuador

Hecho en Ecuador
Made in Ecuador

Este texto ha sido sometido a un proceso de evaluación por pares externos.

Advertencia: "Quedan todos los derechos reservados. Se prohíbe la reproducción, el registro o la transmisión parcial o total de esta obra por cualquier sistema de recuperación de información existente o por existir, sin el permiso previo por escrito del titular de los derechos correspondientes".

Introducción a los Sistemas Computacionales. Una aproximación del estado del arte

AUTORES

Alba Marisol Córdova Vaca[1]

E-mail: alba.cordova@utc.edu.ec

ORCID: https://orcid.org/0000-0002-9134-0750

Jaime Mesias Cajas[4]

E-mail: jaime.cajas@utc.edu.ec

ORCID: https://orcid.org/0000-0002-1852-2187

Geovanny Euclides Silva Peñafiel[2]

E-mail: geovanny.silva1764@utc.edu.ec

ORCID: https://orcid.org/0000-0002-1069-4574

Doris Karina Chicaiza Angamarca[5]

E-mail: doris.chicaiza6508@utc.edu.ec

ORCID: https://orcid.org/0000-0003-1458-8274

Victor Alfonso Cusco Vinueza[3]

E-mail: victor.cusco7756@utc.edu.ec

ORCID: https://orcid.org/0000-0003-2373-2995

Franklin Javier Montaluisa Yugla[6]

E-mail: fjmontaluisa@espe.edu.ec

ORCID: https://orcid.org/0000-0002-9816-1990

[1]Ingeniera en Informática y Sistemas Computacionales; Magister en Evaluación y Auditoria de Sistemas Tecnológicos; Universidad Técnica de Cotopaxi Extensión La Maná.

[2]Ingeniero en Sistemas Informáticos; Magister en Gerencia Informática; Magister en Big Data y Ciencia de Datos; Universidad Técnica de Cotopaxi extensión La Maná.

[3]Ingeniero en Sistemas Computacionales e Informáticos; Magister en Sistemas de Información, Universidad Técnica de Cotopaxi - Extensión La Maná.

[4]Ingeniero en Informática y Sistemas Computacionales; Magister en Sistemas de Información; Universidad Técnica de Cotopaxi.

[5]Ingeniera en informática y sistemas computacionales; Magister en ingeniería de software; Universidad Técnica de Cotopaxi Extensión La Maná.

[6]Ingeniero en Sistemas e Informática; Magister en Ingeniería de Software; Universidad de las Fuerzas Armadas. ESPE.

RESUMEN

El presente texto científico realiza una aproximación sobre el estado del arte relacionado con los Sistemas Computacionales. Posee una estructura que desglosa sus contenidos en introducción, seis capítulos, conclusiones generales y referencias bibliográficas. Los contenidos de los capítulos realizan una descripción sobre: Sistemas computacionales; Internet de las cosas; Principios de la inteligencia artificial; Industria 4.0; Herramientas de ciencia de datos; Seguridad de sistemas computacionales. El texto en su compilación recoge los principales elementos que se relacionan con los sistemas computacionales. Se emplea literatura científica de bases de datos reconocidas con alta actualizada, relevancia y novedad científica.

Palabras clave: sistemas computacionales; internet de las cosas; principios de la inteligencia artificial; industria 4.0; herramientas de ciencia de datos; seguridad de sistemas computacionales.

ABSTRACT

This scientific text makes an approach to the state of the art related to Computational Systems. It has a structure that breaks down its contents into an introduction, six chapters, general conclusions and bibliographical references. The contents of the chapters make a description of: Computer systems; Internet of things; Principles of artificial intelligence; Industry 4.0; data science tools; Security of computer systems. The text in its compilation includes the main elements that are related to computer systems. Scientific literature from recognized databases with high up-to-date, relevance and scientific novelty is used.

Keywords: *computer systems; internet of things; principles of artificial intelligence; industry 4.0; data science tools; security of computer systems.*

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN **1**
CAPÍTULO I: SISTEMAS COMPUTACIONALES **2**
1.1 Introducción del capítulo **2**
1.2 Sistemas computacionales 2
1.3 Hardware del sistema computacional 4
1.4 Software del sistema computacional 5
1.5 Técnicas de procesamiento informático 8
1.5.1. Procesamiento por lotes 8
1.5.2. Spooling 9
1.5.3. Multiprogramación 9
1.5.4. Multiprocesamiento 10
1.5.5. Tiempo compartido 10
1.5.6. Tramitación en línea 10
1.5.7. Procesamiento en tiempo real 10
1.6 Rendimiento de la computadora 11
1.7 Sistemas inteligentes 12
1.8 Computación en la nube 12
1.9 *Green IT* 14
1.10 Conclusiones del capítulo 15
CAPÍTULO II: INTERNET DE LAS COSAS **16**
2.1. Introducción del capítulo 16
2.2. Componentes de IoT 16
2.3. Aplicaciones del IoT 17
2.4. Seguridad en el ecosistema IoT 19
2.5. Conclusiones del capítulo 20
CAPÍTULO III: PRINCIPIOS DE LA INTELIGENCIA ARTIFICIAL **21**
3.1. Introducción del capítulo 21
3.2. Métodos de inteligencia artificial 21
3.2.1. Métodos de solución de problemas 21
3.2.2. Sistemas basados en el conocimiento 22
3.2.3. Redes neuronales artificiales 22
3.2.4. Lógica difusa 22
3.2.5. Computación con palabras 22
3.2.6. Sumarización lingüística de Datos 23
3.2.7. Teoría de los Conjuntos aproximados 23
3.2.8. Metaheurísticas 24
3.2.9. Procesamiento de lenguaje natural 24
3.3. Computación evolutiva (CE) 24
3.3.1. Algoritmos Genéticos (AG): 24

3.3.2. Programación Genética (PG): 25
3.3.3. Programación Evolutiva (PE): 25
3.3.4. Algoritmos de Estimación de Distribuciones (EDA): 25
3.4. Realidad Virtual 25
3.5. Realidad aumentada 26
3.6. Aprendizaje automático 27
3.6.1. Aprendizaje supervisado 28
3.6.2. Aprendizaje no supervisado 28
3.6.3. Aprendizaje semisupervisado 28
3.6.4. Aprendizaje de refuerzo 29
3.6.5. Algoritmos de aprendizaje automático 29
3.7. Aplicaciones de la IA y el aprendizaje automático 38
3.8. Conclusiones del capítulo 40
CAPÍTULO IV: INDUSTRIA 4.0 41
4.1. Introducción del capítulo 41
4.2. Las tecnologías clave de I4.0 41
4.3. Seguridad de la información en la Industria 4.0 44
4.4. Conclusiones del capítulo 45
CAPÍTULO V. HERRAMIENTAS DE CIENCIA DE DATOS 47
5.1 Introducción del capítulo 47
5.2. Tipos de datos del mundo real 47
5.3. Disponibilidad, recopilación, limpieza y gestión de datos 48
5.4. Descubrimiento de conocimiento en bases de datos 50
5.5. Repositorios de datos para la extracción de conocimiento 51
5.6. Minería de datos 51
5.6.1. Problemas de Clasificación 52
5.6.2. Problemas de Regresión 53
5.6.3. Problemas de Clustering 53
5.6.4. Problemas de Reglas de Asociación: 53
5.6.5. Herramientas de minería de datos más utilizadas 54
5.6.6. Minería de datos anómalos 55
5.7. Big Data 55
5.8. Conclusiones del capítulo 56
CAPÍTULO VI: SEGURIDAD DE SISTEMAS COMPUTACIONALES 57
6.1. Introducción del capítulo 57
6.2. Vulnerabilidades de los sistemas computacionales 57
6.3. Categorías de ataque 58
6.4. Protocolos de defensa 61
6.5. Sistemas de Detección de Intrusos 63
6.6. Blockchain 64
6.7. Conclusiones del capítulo 65
CAPÍTULO VII: INGENIERÍA DE SOFTWARE 67
7.1 Introducción 67
7.2 Ingeniería de software 67

7.3 Teorías que sustentan los modelos de ciclo de vida de software 72
7.4 Teorías más influyentes en el desarrollo de software y los modelos de ciclo de vida: 72
7.5 Metodologías de desarrollo de software. 73
7.6 Diseño de software 77
7.7 Principios de diseño de software 77
7.8 Cómo aplicar los principios para diseñar software de alta calidad. 78
7.9 Mantenimiento del software 78
7.10 Herramientas de ingeniería de software 79
7.11 Cómo se utilizan las herramientas en la práctica 88
7.12 Pruebas de software 90
7.13 Mantenimiento de software 92
7.14 Conclusiones del capítulo 94
CONCLUSIONES GENERALES 96
REFERENCIAS BIBLIOGRÁFICAS 97

INTRODUCCIÓN

No pasa una década sin que se produzca un cambio disruptivo en el dominio de los Sistemas Computacionales. Cada cambio trae consigo nuevas actualizaciones de hardware y optimizaciones de software que impulsan a las organizaciones a desarrollar sus prácticas de desarrollo que integren los aspectos novedosos del dominio: Informática personal; el internet; la web; Informática móvil; computación en la nube, Internet de las Cosas (IoT), Aprendizaje automático.

Las instituciones se enfrentan a un conjunto de plataformas y tecnologías en constante cambio que deben escalar para hacer más eficientes sus procesos. Algunos ingenieros aprenden nuevos métodos y técnicas en la escuela y los llevan a las organizaciones para las que trabajan. Otros aprenden nuevas habilidades en el trabajo o en forma paralela, ya que anticipan la necesidad de talento latente de su organización.

Una de las tendencias con más aplicaciones en la industria del software es la integración de capacidades de Inteligencia Artificial (IA) basadas en los avances en el Aprendizaje Automático (ML, *Machine Learning*). Los avances recientes en el aprendizaje automático han estimulado un interés generalizado dentro del sector de la tecnología de la información sobre la integración de capacidades de IA en software y servicios. Asimismo, la integración del Internet de las Cosas (IoT) con la Computación en la Nube (*Cloud Computing*) ha alcanzado resultados exponenciales en el sector industrial y doméstico.

La Industria 4.0 es típicamente la automatización continua de las prácticas industriales y de fabricación convencionales, incluido el procesamiento exploratorio de datos, utilizando nuevas tecnologías inteligentes como la automatización del aprendizaje automático. Analizar las actualizaciones de la Industria 4 constituye una de las vías para actualizar las principales actualizaciones en los sistemas computacionales y de la Inteligencia artificial como factor integrador.

En este contexto, dentro de la IA se han creado técnicas como la Sumarización Lingüística de Datos (SLD) que permiten identificar las relaciones subyacentes en los datos, tales como: correlaciones, tendencias, agrupaciones o anomalías. Estas técnicas se caracterizan por generar resúmenes lingüísticos a partir de bases de datos estructuradas utilizando lenguaje natural facilitando por su naturaleza la toma de decisiones.

Es cierto que se han creado modelos, algoritmos y sistemas que se encargan de soportar la gestión de los recursos humanos, utilizando para su solución elementos muy actuales como la Minería de Datos (DM), Inteligencia Artificial, *Soft Computing* y la computación con palabras entre otros. Sin embargo, un elemento que no se debe descuidar ya que está indisolublemente unido a las tecnologías, es la Seguridad de los sistemas computacionales.

Teniendo en cuenta estas ideas; el texto que se pone a disocian del Lector "*INTRODUCCIÓN A LOS SISTEMAS COMPUTACIONALES. UNA APROXIMACIÓN DEL ESTADO DEL ARTE*" analiza las tendencias actuales de los sistemas computacionales y sus aplicaciones para el desarrollo actual.

CAPÍTULO I: SISTEMAS COMPUTACIONALES

1.1 Introducción del capítulo

Las computadoras modernas son electrónicas y procesan información digital. La máquina física consta de transistores, circuitos digitales implementados con transistores, cables y componentes mecánicos en dispositivos periféricos utilizados para el almacenamiento de información. Estas entidades físicas se denominan colectivamente hardware. Los programas de sistema y de aplicación se denominan software (S. Liu et al., 2019).

Un sistema computacional de propósito general es una máquina programable que puede resolver problemas aceptando entradas e instrucciones sobre cómo usar estas entradas. Las instrucciones están incluidas en el software que normalmente contienen secuencias de ellas. Numerosos sistemas computacionales han sido diseñados y construidos para ayudar a los seres humanos en el procesamiento de información y cálculos numéricos (Sittón-Candanedo et al., 2019). Como resultado, han surgido varios modelos en el campo del diseño de sistemas informáticos. Estos modelos difieren en la arquitectura de los procesadores, el modelo subyacente de computación, la arquitectura de la memoria principal o las técnicas utilizadas para interconectar los recursos básicos dentro de la computadora (F. Liu et al., 2019).

Este capítulo presenta un resumen de los modelos informáticos fundamentales. Se aborda la tarea de análisis del rendimiento de los sistemas informáticos. Se presentan los fundamentos del diseño y organización, y se describe el procedimiento convencional para la ejecución de programas informáticos. También se incluye una descripción general de las principales características e interacciones de los componentes de hardware y software de los sistemas computacionales modernos.

1.2 Sistemas computacionales

Un sistema computacional se define como un conjunto de dispositivos informáticos integrados que toman, generan, procesan y almacenan datos e información. En términos generales, un sistema computacional se construye con al menos un dispositivo informático (Sebastian et al., 2020). En la literatura, tanto los sistemas informáticos de un solo dispositivo como los sistemas distribuidos que consisten en un conjunto de dispositivos informáticos han sido ampliamente explorados.

> Los objetivos de investigación en sistemas computacionales incluyen: el **rendimiento**, la **eficiencia energética**, la **fiabilidad** y la **seguridad**.

Un sistema informático comprende principalmente una Unidad Central de Procesamiento (CPU), memoria, dispositivos de entrada/salida y dispositivos de almacenamiento. Todos estos componentes funcionan juntos como una sola unidad para entregar el resultado deseado. Un sistema informático viene en varias formas y tamaños. Puede variar desde un servidor de alta gama hasta una computadora de escritorio personal, una computadora portátil, una tableta o un teléfono inteligente (Abbasi & Rafiee, 2020).

Un sistema informático generalmente incluye, además del procesador, un teclado para escribir comandos, un mouse o teclado o joystick para posicionar las entradas del menú, un monitor para mostrar la información que ha producido el sistema informático, memoria para almacenar información temporalmente, discos y Memorias USB de uno u otro tipo para almacenar información durante mucho tiempo, incluso después de que la computadora haya sido apagada, conexiones a otros dispositivos como una impresora para obtener copias en papel de esa información y la colección de programas (el software) que el usuario desea ejecutar (Xiao et al., 2019). Un sistema computacional está compuesto por:

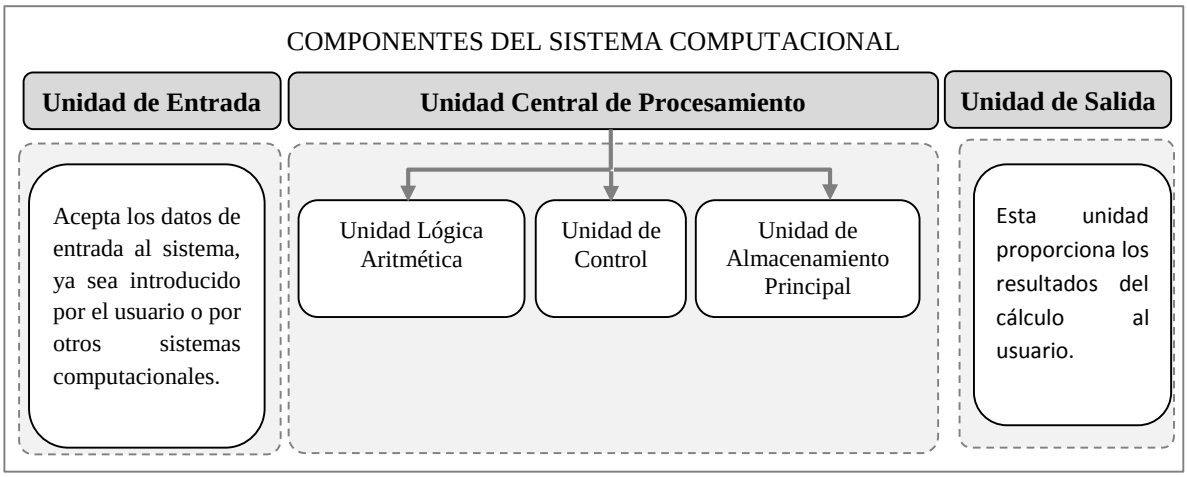


Figura 1. Componentes del sistema computacional.

Unidad de entrada: Los dispositivos de entrada son los dispositivos que se utilizan para alimentar programas y datos a la computadora. El sistema de entrada conecta el entorno externo con el sistema informático. Los dispositivos de entrada son los medios de comunicación entre el usuario y el sistema informático. Los dispositivos de entrada típicos incluyen el teclado, los disquetes, el mouse, el micrófono, el lápiz óptico, el joystick, las cintas magnéticas (Yang et al., 2020). La forma en que los datos ingresan a la computadora a través de cada uno de estos dispositivos es diferente. Sin embargo, una computadora puede aceptar datos solo en una forma específica. Por lo tanto, estos dispositivos de entrada transforman los datos que se les alimentan en una forma que puede ser aceptada por la computadora (Ruan & Li, 2020). Estos dispositivos son un medio de comunicación y una estación entre el usuario y los sistemas informáticos. Así, las funciones de la unidad de entrada son:

- Aceptar información (datos) y programas.
- Convertir los datos en una forma que la computadora pueda aceptar.
- Proporcionar estos datos convertidos a la computadora para su posterior procesamiento.

Unidad Central de Procesamiento: Este es el cerebro de cualquier sistema informático. La Unidad Central de Procesamiento (CPU) se compone de tres partes:

- **La Unidad de Control:** La Unidad de Control controla las operaciones de todo el sistema informático. La unidad de control obtiene las instrucciones de los programas almacenados en la unidad de almacenamiento principal, interpreta estas instrucciones y, posteriormente, dirige a las otras unidades para que ejecuten las instrucciones. Así gestiona y coordina todo el sistema informático (Lu, 2019).

- **La Unidad Lógica Aritmética:** La Unidad Lógica Aritmética (ALU) en realidad ejecuta las instrucciones y realiza todos los cálculos y decisiones. Los datos se mantienen en la unidad de almacenamiento principal y se transfieren a la ALU siempre que sea necesario. Los datos se pueden mover del almacenamiento primario a la unidad lógica aritmética varias veces antes de que se complete todo el procesamiento. Después de la finalización, los resultados se envían a la sección de almacenamiento de salida y los dispositivos de salida (Norouzi et al., 2020).
- **La Unidad de Almacenamiento Principal:** También se conoce como memoria principal. Antes de que comience el procesamiento real, los datos y las instrucciones enviadas a la computadora a través de las unidades de entrada se almacenan en esta unidad de almacenamiento principal. De manera similar, los datos que van a salir del sistema informático también se almacenan temporalmente en la memoria primaria. También es el área donde se almacenan los resultados intermedios de los cálculos (Sebastian et al., 2020). La memoria principal tiene la sección de almacenamiento que contiene los programas de computadora durante la ejecución. Así la unidad primaria:
    a. Almacena datos y programas durante el procesamiento real.
    b. Almacena los resultados temporales del procesamiento intermedio.
    c. Almacena los resultados de la ejecución temporalmente.

Unidad de salida: Los dispositivos de salida dan los resultados del proceso y los cálculos al mundo exterior. Las unidades de salida aceptan los resultados producidos por la computadora, los convierten en una forma legible por humanos y los proporcionan a los usuarios. Los dispositivos de salida más comunes son impresoras, trazadores, pantallas de visualización, unidades de cinta magnética, etc. Una computadora es un sistema de procesamiento de datos rápido y preciso que acepta datos, realiza varias operaciones en los datos, tiene la capacidad de almacenar datos y procesarlos con el conjunto de instrucciones que se le dan (Putro et al., 2020). Los datos son la información proporcionada por el usuario a la computadora y el conjunto de instrucciones para realizar la operación sobre los datos es el programa de la computadora.

1.3 Hardware del sistema computacional

El hardware de la computadora son las partes físicas de la máquina, como el monitor, el teclado, los discos; mientras que el software son los diversos programas, procedimientos y otra documentación que se utiliza para operar el hardware de manera eficiente (Efthymiou et al., 2021). La evolución o desarrollo de las computadoras se caracteriza por las distintas generaciones de computadoras.

I. La primera generación tenía máquinas muy grandes y complejas que hacían uso de la tecnología de tubos de vacío.
II. La invención de los transistores en la segunda generación redujo el tamaño de las computadoras y surgió el concepto de programa almacenado, así como también se introdujeron lenguajes de mayor nivel.
III. La tercera generación se caracterizó por los Circuitos Integrados y la producción comercial de computadoras.

IV. La cuarta generación vio la invención de las microcomputadoras con integración a muy gran escala, redes e introducción de computadoras personales.

V. La quinta generación o la actual ha visto avances en el procesamiento paralelo y las tecnologías de superconductores.

Desde el punto de vista del hardware, se supone convenientemente que una computadora tiene una jerarquía de cinco niveles. Los cinco niveles corresponden a

1. Puertos de red para conectarse con el mundo exterior. Estos puertos pueden no estar necesariamente disponibles, ya que una computadora puede ser una máquina de computación y/o procesamiento de información independiente.
2. Dispositivos periféricos o de almacenamiento masivo para almacenamiento de datos y programas.
3. Memoria principal.
4. Cachés de datos y programas: memorias rápidas para recuperar datos por contenido.
5. Unidad Central de Procesamiento (CPU o procesador).

Unidad de Procesamiento de Gráficos (GPU)

La Unidad de Procesamiento de Gráficos (GPU, *Graphic Process Unit*) se ha convertido en una parte integral de los principales sistemas informáticos actuales. Durante los últimos años, ha habido un marcado aumento en el rendimiento y las capacidades de las GPU (Seritan et al., 2021). La GPU moderna no es solo un potente motor de gráficos, sino también un procesador programable altamente paralelo que presenta una aritmética máxima y un ancho de banda de memoria que supera sustancialmente a su contraparte de CPU (Ni et al., 2019).

Las GPU se han utilizado para acelerar el rendimiento de diversas aplicaciones, como la simulación de partículas, el modelado molecular y el procesamiento de imágenes. La arquitectura GPU está basada en el paralelismo masivo, proporcionando cientos de cores (elementos de procesamiento), donde cada core es un procesador pipeline multi-etapa. Los cores se agrupan para generar un multiprocesador (SM) de caracter SIMD al nivel de grupos de 32 threads. Cada SM tiene su propio conjunto de registros y memoria local, y los cores dentro del mismo SM tienen memoria compartida limitada (Keith et al., 2022).

El rápido aumento de la capacidad de programación y la capacidad de la GPU ha generado una comunidad de investigación que ha mapeado con éxito una amplia gama de problemas complejos y exigentes desde el punto de vista computacional de la GPU (Partel et al., 2019). Este esfuerzo en computación de propósito general en la GPU, ha posicionado a la GPU como una alternativa convincente a los microprocesadores tradicionales en los sistemas informáticos de alto rendimiento del futuro. Se destacan éxitos de computación de GPU en física de juegos y biofísica computacional que brindan ganancias de rendimiento de orden de magnitud en comparación con aplicaciones de CPU optimizadas (Miscuglio & Sorger, 2020).

1.4 Software del sistema computacional

El software que proporciona la funcionalidad básica para operar una computadora al interactuar directamente con su hardware constituyente se denomina software del sistema. Un software de

sistema sabe cómo operar y usar diferentes componentes de hardware de una computadora. Proporciona servicios directamente al usuario final o a algún otro software (Neese, 2022). Los ejemplos de software del sistema incluyen sistemas operativos, utilidades del sistema, controladores de dispositivos, etc.

Software de aplicación: El software de aplicación es un programa o un conjunto de programas que se escriben para llevar a cabo una aplicación específica: nómina, contabilidad financiera, etc. Hoy en día se encuentran disponibles paquetes o software de aplicaciones especiales para áreas especializadas como dibujo, ingeniería, fabricación, banca y publicación. El conjunto de programas que juntos forman un paquete de aplicación se denominan programas de aplicación (Cicchetti et al., 2019).

El software del sistema proporciona la funcionalidad principal del sistema informático. Sin embargo, diferentes usuarios necesitan el sistema informático para diferentes propósitos dependiendo de sus requisitos. Por lo tanto, se necesita una nueva categoría de software para satisfacer los diferentes requisitos de los usuarios finales. Este software específico que funciona sobre el software del sistema se denomina software de aplicación (Cicchetti et al., 2019). Nuevamente, existen dos amplias categorías de software de aplicación: software de aplicación de propósito general y personalizado:

- Software de uso general: El software de aplicación desarrollado para aplicaciones genéricas, para atender a un público más amplio en general, se denomina software de propósito general. Los usuarios finales pueden utilizar dicho software de aplicación listo para usar según sus requisitos. Por ejemplo, cualquier usuario de computadora puede usar la herramienta de hoja de cálculo Calc de LibreOffice para hacer cálculos o crear una hoja de cuenta. Adobe Photoshop, GIMP, el navegador web Mozilla, iTunes, etc., entran en la categoría de software de propósito general (Pham et al., 2020).
- Software personalizado: Se trata de software de aplicación personalizado o hecho a la medida, que se desarrollan para cumplir con los requisitos de una organización específica o un individuo. Se adaptan mejor a las necesidades de un individuo o de una organización, considerando que están diseñados según requisitos especiales. Algunos ejemplos de software definido por el usuario incluyen sitios web, software de gestión académica, software de contabilidad (Hölldobler et al., 2019).

Software del sistema: El software del sistema controla el funcionamiento del sistema informático. Ayuda al usuario a usar la computadora al permitirle comunicarse con el sistema. El software del sistema controla el funcionamiento de otros software, hardware, dispositivos de hardware como impresoras, memoria, CPU, etc (Kossiakoff et al., 2020). Por lo tanto, hacen que la operación de la computadora sea más eficiente. Los programas incluidos en el software del sistema son programas de sistemas. Sin los programas de sistemas, no sería posible que los programas de aplicación funcionaran en la computadora. El software de los sistemas generalmente lo proporciona el fabricante del hardware de la computadora. Sin el software de los sistemas, el hardware no funcionaría (Gleirscher & Marmsoler, 2020). Existen seis clases fundamentales de software de sistema:

1. **Procesadores de lenguaje:** Traducen programas escritos en lenguajes simbólicos en código legible por máquina. Existen dos tipos básicos de procesadores de lenguaje, a saber: compiladores y ensambladores. Los compiladores asumen programas escritos en lenguajes de alto nivel. Los ensambladores traducen programas escritos en lenguajes ensambladores.
2. **Rutinas de programas de biblioteca:** Pueden ser invocadas por programas de aplicación para realizar tareas específicas, evitando así la recodificación para tareas ampliamente utilizadas y facilitando la programación.
3. **Cargadores**: Almacenan código legible por máquina en la memoria de la computadora para su ejecución.
4. **Enlazadores:** Combinan códigos de objeto de diferentes rutinas para formar un solo código ejecutable para el cargador.
5. **Sistema operativo:** Administra los recursos de la computadora de manera efectiva en tiempo de ejecución.
6. **Monitoreo**: Programas que monitorean las actividades de la computadora.

Sistemas operativos

Como su nombre lo indica, el sistema operativo es un software de sistema que opera la computadora. Un sistema operativo es el software de sistema más básico, sin el cual otro software no puede funcionar (Pan et al., 2021). El sistema operativo administra otros programas de aplicación y proporciona los sistemas operativos populares Windows, Linux, Macintosh, Ubuntu, Fedora, Android, iOS, etc.

Se puede considerar que un sistema operativo (OS, por sus siglas en inglés) es un administrador de recursos que administra todos los recursos de una computadora, es decir, su hardware, incluida la CPU, la RAM, el disco, la red y otros dispositivos de entrada y salida. También controla varias aplicaciones de software y controladores de dispositivos, administra la seguridad del sistema y maneja el acceso de diferentes usuarios. Es el software de sistema más importante (Bauwens et al., 2020).

Los objetivos principales de un sistema operativo son dos. El primero es proporcionar servicios para construir y ejecutar programas de aplicación. Cuando es necesario ejecutar un programa de aplicación, es el sistema operativo el que carga ese programa en la memoria y lo asigna a la CPU para su ejecución (Paccagnella et al., 2020). Cuando es necesario ejecutar varios programas de aplicación, el sistema operativo decide el orden de ejecución.

El segundo objetivo de un sistema operativo es proporcionar una interfaz al usuario a través de la cual el usuario puede interactuar con la computadora. Una interfaz de usuario es un componente de software que forma parte del sistema operativo y cuyo trabajo es recibir comandos o entradas de un usuario para que el sistema operativo los procese (Narayanan et al., 2019).

Los Sistemas Operativos proporcionan la plataforma adecuada para la ejecución de programas de aplicación mediante la gestión eficaz de los recursos informáticos. Un sistema operativo se define como un conjunto integrado de programas que gestionan los diversos recursos y operaciones de un sistema informático (Ganesan et al., 2021). Muchas de las rutinas del OS residen en la memoria y se invocan según sea necesario:

- Asignan espacio de memoria y disco a programas de usuario en tiempo estático y de ejecución.

- El sistema operativo es el que tiene que decidir y dar prioridad a los trabajos que se van a ejecutar.
- Controlan dispositivos periféricos reconociendo comandos de entrada o enviando datos/comandos a dispositivos de salida.
- Se encargan de maximizar la utilización de dispositivos que están más cerca de la CPU para un mejor rendimiento.
- Asignan privilegios a los programas de usuario para acceder a datos/rutinas/dispositivos, identificar componentes defectuosos y verificar resultados erróneos.
- Para computadoras muy avanzadas, el OS con entornos informáticos multiusuario, protege los programas y datos del usuario de interferencias y asigna recursos a los programas del usuario de manera equitativa y compartida.

Utilidades del sistema

El software utilizado para el mantenimiento y la configuración del sistema informático se denomina utilidad del sistema. Algunas utilidades del sistema se envían con el sistema operativo, por ejemplo, la herramienta de desfragmentación del disco, la utilidad de formato, la utilidad de restauración del sistema, etc. Otro conjunto de utilidades son aquellas que no se envían con el sistema operativo pero que son necesarias para mejorar el rendimiento del sistema, por ejemplo, software antivirus, herramienta de limpieza de disco, software de compresión de disco (Englander & Wong, 2021).

Controladores de dispositivos

El propósito de un controlador de dispositivo es garantizar el correcto funcionamiento de un dispositivo en particular. Cuando se trata del funcionamiento general de un sistema informático, el sistema operativo hace el trabajo (Macenski et al., 2022). Pero todos los días se agregan nuevos dispositivos y componentes a un sistema informático. No es posible que el sistema operativo solo opere todos los dispositivos existentes y nuevos, donde cada dispositivo tiene características diversas. La responsabilidad del control general, la operación y la gestión de un dispositivo en particular a nivel de hardware se delega a su controlador de dispositivo (Bai et al., 2019).

El controlador del dispositivo actúa como una interfaz entre el dispositivo y el sistema operativo. Proporciona los servicios necesarios ocultando los detalles de las operaciones realizadas a nivel de hardware del dispositivo. Al igual que un traductor de idiomas, un controlador de dispositivo actúa como mediador entre el sistema operativo y el dispositivo conectado (Malallah et al., 2021).

1.5 Técnicas de procesamiento informático

1.5.1. Procesamiento por lotes

En el procesamiento por lotes, el operador recopila una cantidad de trabajos fuera de línea y cuando se recopila un lote de trabajos, se ingresan a la computadora para su procesamiento. A continuación, los trabajos se procesan sin la intervención del usuario (Yoo et al., 2021). Las aplicaciones típicas del procesamiento por lotes incluyen aplicaciones como nómina,

preparación de facturas, etc., donde la información no debe actualizarse con frecuencia. Las desventajas del procesamiento por lotes son:

- Los datos tienen que ser primero acumulados y luego procesados. Por lo tanto, existe la posibilidad de retraso en la ejecución de los trabajos.
- No es posible cambiar la prioridad de los trabajos. Si ambos trabajos tienen la misma prioridad, uno de ellos debe esperar en el lote hasta la ejecución completa del otro.

1.5.2. Spooling

Spooling se utiliza para mejorar la velocidad de procesamiento del procesamiento por lotes. Spooling significa salida periférica simultánea en línea. El proceso de almacenamiento de datos de entrada y resultados de salida en almacenamiento secundario se conoce como spooling. Los datos de entrada se almacenan en discos magnéticos y se alimentan a la CPU cuando no está demasiado ocupada. Luego, el procesador procesa los datos y la salida resultante se almacena nuevamente en un dispositivo de almacenamiento secundario. Así, en el spooling, el medio de almacenamiento magnético actúa como un búfer entre la memoria y los dispositivos de entrada/salida.

En muchas computadoras se proporcionan procesadores de entrada/salida de propósito especial que pueden funcionar independientemente de la CPU, dejando así libre al procesador principal una vez que finaliza la ejecución del programa. Los procesadores de E/S luego imprimen los datos en cola desde la cinta o el disco en la impresora. Por lo tanto, el spooling mejora la eficiencia del uso de la memoria y la CPU.

1.5.3. Multiprogramación

En multiprogramación la CPU es capaz de ejecutar más de un programa al mismo tiempo. Más de un programa puede residir en la memoria principal en un momento dado, sin embargo, el procesador solo puede ejecutar una instrucción a la vez. La velocidad de funcionamiento de la CPU es mucho más rápida que la de los dispositivos de entrada y salida. Por lo tanto, cuando un programa está ocupado con operaciones de entrada y salida, la CPU puede asignar tiempo a otros programas en lugar de permanecer inactivo. Por lo tanto, varios usuarios pueden compartir el tiempo de la CPU.

Varios programas pueden residir en la memoria principal del sistema informático. El lugar en la memoria donde reside un programa se conoce como partición. Dependiendo del sistema operativo, se decide el número real de particiones en la memoria y el número de programas que pueden residir simultáneamente. Así, en la multiprogramación es posible ejecutar varios programas en mucho menos tiempo del que se necesitaría para ejecutarlos uno tras otro.

La multiprogramación aumenta tanto el rendimiento como el tiempo de respuesta del sistema informático. Sin embargo, se requiere que los sistemas operativos que pueden admitir la multiprogramación tengan una gran capacidad de memoria y, al mismo tiempo, un mecanismo eficaz para proteger la memoria.

1.5.4. Multiprocesamiento

En los sistemas de multiprocesamiento, dos o más CPU están conectadas entre sí. Por lo tanto, es posible ejecutar instrucciones de diferentes programas al mismo tiempo. Por lo tanto, se puede ejecutar más de una instrucción simultáneamente. Se reduce el tiempo de inactividad de la computadora, ya que no hay intervención del usuario entre trabajos (Hunt & Hunt, 2019).

Ejecutar diferentes instrucciones de un mismo programa. Diferentes sistemas de multiprocesamiento utilizan diferentes tipos de configuraciones de memoria. Algunos sistemas tienen una memoria principal común para todas las CPU, en algunos sistemas cada sistema tiene su propia memoria principal, mientras que en otros cada CPU puede tener su propia memoria y compartir una memoria común con otros procesadores. Los sistemas de multiprocesamiento requieren un sistema operativo muy complejo y sofisticado para coordinar todas las actividades de las múltiples CPU y otros dispositivos. Los sistemas de multiprocesamiento también son muy caros (Smith & Smith, 2022).

1.5.5. Tiempo compartido

En el tiempo compartido, es posible que varios usuarios ejecuten más de una aplicación al mismo tiempo en la computadora. Esto se logra proporcionando un terminal separado para cada usuario. Todos estos terminales están conectados a la computadora principal. El tiempo de GPU se divide entre todos los usuarios de forma programada. El tiempo que obtiene cada usuario se denomina segmento de tiempo (Wu et al., 2022). La CPU cambia de un usuario a otro y ejecuta una parte del proceso en el intervalo de tiempo. Este proceso continúa hasta que se ejecuta el trabajo. En el tiempo compartido, como en la multiprogramación, solo un programa tiene el control de la CPU en un momento dado (Zainab et al., 2021).

En un entorno de tiempo compartido, no siempre es posible que todos los programas de todos los usuarios residan en la memoria principal. Solo el programa de control y algunos programas residen en la memoria principal. Los programas restantes se cargan desde la memoria secundaria a medida que se van a ejecutar.

1.5.6. Tramitación en línea

En el procesamiento en línea, la CPU envía directamente los datos de la transacción a los dispositivos secundarios de almacenamiento en línea sin clasificarlos, desde el punto donde se generan los datos. El acceso y recuperación de datos es muy rápido. En estos sistemas, los dispositivos están conectados directamente a la CPU para entrada o consulta.

1.5.7. Procesamiento en tiempo real

El procesamiento en tiempo real es un sistema de procesamiento en línea donde los registros se actualizan y los datos se procesan tan pronto como se realiza la transacción. Los sistemas en tiempo real permiten a los usuarios comunicarse con la computadora durante el procesamiento mismo. En los sistemas en tiempo real, varios terminales pueden estar conectados a una única CPU. Un número de estaciones remotas están conectadas a la computadora con líneas de comunicación y varias estaciones operan simultáneamente. Los registros de actualización de

transacciones. En el procesamiento fuera de línea, el procesamiento de datos no está controlado directamente por la CPU (Fu & Cui, 2019). En las minicomputadoras, se utilizan varias terminales para ingresar los datos en un almacenamiento secundario, como una cinta o un disco. Los datos se validan y luego se ingresan en la CPU principal en modo por lotes para su procesamiento (Zheng et al., 2019).

1.6 Rendimiento de la computadora

Algunos sistemas informáticos tienen un mejor rendimiento que otros para algunas aplicaciones. El rendimiento de una computadora generalmente varía con el dominio de la aplicación (Ding et al., 2019). El rendimiento se mide de acuerdo con:

- El tiempo de respuesta del programa.
- El rendimiento de la máquina.

Se deben usar aplicaciones reales o puntos de referencia para evaluar el rendimiento de la computadora. Los puntos de referencia son programas sintéticos típicos de la carga de trabajo esperada. Estos puntos de referencia son diferentes para diferentes dominios de aplicación. El rendimiento real de una computadora para un programa determinado depende de:

- El comportamiento del programa: las dependencias de instrucción en el programa.
- La arquitectura del procesador: el tamaño de sus conductos y el tamaño del chip integrado, la caché.
- La arquitectura del sistema completo.
- La tecnología de silicio utilizada

Una dependencia de instrucción puede asumir la forma de una dependencia de datos, recursos o control. Dos instrucciones dependen de los datos si usan la misma variable como entrada y/o salida. Dependen de los recursos si usan el mismo recurso de hardware y dependen del control si su orden de ejecución no se puede determinar en tiempo estático/compilación. Un número reducido de dependencias generalmente implica un mejor rendimiento.

Para una arquitectura de computadora determinada, el rendimiento aumenta con los aumentos de la frecuencia del reloj, las reducciones de CPI (*cycles per instruction*) a través de mejoras en el procesador o el recuento de instrucciones o las reducciones de CPI debido a las mejoras del compilador (Leiserson et al., 2020).

> La medida de CPI se define como la relación del número acumulado de ciclos de reloj para todas las instrucciones en el conjunto de instrucciones, dividido por el número total de instrucciones.

Los requisitos computacionales son grandes. La representación en tiempo real requiere miles de millones de píxeles por segundo y cada píxel requiere cientos o más operaciones. Las GPU deben ofrecer una enorme cantidad de rendimiento informático para satisfacer la demanda de aplicaciones complejas en tiempo real. El paralelismo es sustancial. Afortunadamente, la canalización de gráficos es adecuada para el paralelismo. Las operaciones en vértices y fragmentos se adaptan bien a las unidades de computación paralelas programables estrechamente acopladas y de grano fino, que a su vez son aplicables a muchos otros dominios computacionales.

El rendimiento es más importante que la latencia. Las implementaciones de GPU de la canalización de gráficos priorizan el rendimiento sobre la latencia. El sistema visual humano opera en escalas de tiempo de milisegundos, mientras que las operaciones dentro de un procesador moderno toman nanosegundos. Esta brecha de seis órdenes de magnitud significa que la latencia de cualquier operación individual no es importante. Como consecuencia, la canalización de gráficos es bastante profunda, quizás de cientos a miles de ciclos, con miles de primitivas en vuelo en un momento dado.

1.7 Sistemas inteligentes

Los sistemas inteligentes (SI) proporcionan un enfoque metodológico estandarizado para resolver problemas importantes y bastante complejos y obtener resultados consistentes y confiables a lo largo del tiempo. Desde la perspectiva de la computación, la inteligencia de un sistema se puede caracterizar por su flexibilidad, adaptabilidad, memoria, aprendizaje, dinámica temporal, razonamiento y la capacidad de manejar información incierta e imprecisa (Pardo et al., 2022).

Los SI están motivados por la necesidad de resolver problemas complejos mejorando la eficiencia. Es un sistema que emula algunos aspectos de la inteligencia exhibidos por la naturaleza. Estos incluyen el aprendizaje, la adaptabilidad, la solidez en los dominios de los problemas, la mejora de la eficiencia, la compresión de la información y el razonamiento extrapolado (Jahan et al., 2020).

Los sistemas inteligentes de aprendizaje automático actuales se basan en el rendimiento: la atención se centra en la precisión predictiva/clasificación, en función de las propiedades conocidas aprendidas de las muestras de entrenamiento. La mayoría de los modelos no paramétricos basados en aprendizaje automático requieren un alto costo computacional para encontrar los óptimos globales (Ntoutsi et al., 2020). Con la tarea de aprendizaje en un gran conjunto de datos, la cantidad de nodos ocultos dentro de la red aumentará significativamente, lo que finalmente conducirá a un aumento exponencial de la complejidad computacional.

1.8 Computación en la nube

La computación en la nube (*Cloud Computing*) se puede clasificar como un nuevo paradigma para el aprovisionamiento dinámico de servicios informáticos respaldados por centros de datos de última generación que generalmente emplean tecnologías de Máquinas Virtuales (VM) con fines de consolidación y aislamiento del entorno (Bello et al., 2021). La computación en la nube ofrece una infraestructura, una plataforma y un software (aplicaciones) como servicios que se ponen a disposición de los consumidores en un modelo de pago por uso. En la industria, estos servicios se denominan Infraestructura como Servicio (IaaS), Plataforma como servicio (PaaS) y Software como Servicio (SaaS), respectivamente. Muchos proveedores de servicios de computación, incluidos Google, Microsoft, Yahoo e IBM, están implementando rápidamente centros de datos en varios lugares del mundo para brindar servicios de computación en la nube (Sunyaev & Sunyaev, 2020).

La computación en la nube juega un papel importante e interconectado entre las tecnologías de tendencia actuales relacionadas con el Internet de las cosas (IoT), el Aprendizaje Automático (ML), el aprendizaje profundo (DL) y los sistemas ciberfísicos (CPS) (Sadeeq et al., 2021).

Todos estos dominios principales requieren plataformas en la nube como plataformas obligatorias para el almacenamiento y procesamiento de datos de sensores y dispositivos y para brindar posibles sugerencias.

Los aspectos esenciales de la computación en la nube se han informado en la definición proporcionada por el Instituto Nacional de Estándares y Tecnologías (NIST):

> *"La computación en la nube es un modelo para permitir el acceso a la red ubicuo, conveniente y bajo demanda a un grupo compartido de recursos informáticos configurables (p. ej., redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y liberar rápidamente con un esfuerzo de gestión mínimo o interacción con el proveedor de servicios".*

En los últimos años, la aparición de la computación en la nube ha tenido un gran impacto en la industria de TI. La disponibilidad de almacenamiento virtualmente ilimitado y capacidades de procesamiento a bajo costo permitió la realización de un nuevo modelo informático, en el que los recursos virtualizados se pueden arrendar bajo demanda, proporcionándose como utilidades generales. Las grandes empresas (como Amazon, Google, Facebook, etc.) adoptaron ampliamente este paradigma para la prestación de servicios a través de Internet, obteniendo beneficios tanto económicos como técnicos (Zhang et al., 2020).

Cloud Computing es una tecnología disruptiva con profundas implicaciones para la prestación de servicios de Internet, así como para el sector de TI en su conjunto. Sin embargo, aún quedan por resolver varios problemas técnicos y comerciales. Se han identificado problemas específicos para cada modelo de servicio, que están principalmente relacionados con la seguridad (p. ej., seguridad e integridad de los datos, seguridad de la red), privacidad (p. ej., confidencialidad de los datos) y acuerdos de nivel de servicio, que podrían ahuyentar a parte de los usuarios potenciales (Sriram, 2022).

Además, la falta de API estándar evita que los clientes extraigan fácilmente código y datos de un sitio para ejecutarlos en otro. Más en general, al subcontratar la infraestructura a un proveedor de la nube, los clientes de la nube pública están necesariamente expuestos a aumentos de precios, problemas de confiabilidad o incluso a la quiebra de los proveedores. Las principales características que lo convierten en un modelo exitoso, su arquitectura en capas y los modelos de servicio estándar (Qi & Tao, 2019).

Aplicaciones del *Cloud computing*

Arquitectura en capas y modelos de servicio: La arquitectura de la nube se puede dividir en cuatro capas:

1. Centro de datos (hardware).
2. Infraestructura.
3. Plataforma.
4. Aplicación.

Cada uno de ellos puede verse como un servicio para la capa superior y como un consumidor para la capa inferior.

En la práctica, los servicios en la nube se pueden agrupar en tres categorías principales: software como servicio (**SaaS**), plataforma como servicio (**PaaS**) e infraestructura como servicio (**IaaS**).

- **SaaS** se refiere al aprovisionamiento de aplicaciones que se ejecutan en entornos de nube. Por lo general, se puede acceder a las aplicaciones a través de un cliente ligero o un navegador web.
- **PaaS** se refiere a los recursos de la capa de la plataforma (por ejemplo, soporte del sistema operativo, marcos de desarrollo de software, etc.).
- **IaaS** se refiere a proporcionar recursos de procesamiento, almacenamiento y red, lo que permite al consumidor controlar el sistema operativo, el almacenamiento y las aplicaciones. Ha suscitado el mayor interés hasta el momento.

Tipos de Nubes: Se han identificado diferentes tipos de Nubes en la literatura, como se informa a continuación:

a. **Nube Privada:** aprovisionada para uso exclusivo de una sola organización, típicamente propiedad, administrada y operada por la propia organización;
b. **Community Cloud:** aprovisionado para uso exclusivo de una comunidad específica de consumidores que tienen preocupaciones compartidas;
c. **Nube pública**: aprovisionada para uso abierto por parte del público en general;
d. **Nube híbrida:** composición de dos o más infraestructuras de nube distintas (privada, comunitaria o pública);
e. **Nube privada virtual:** alternativa destinada a abordar los problemas relacionados con las nubes públicas y privadas, aprovechando las tecnologías de red privada virtual (VPN) para permitir que los propietarios de negocios configuren las configuraciones de red requeridas.

Teniendo en cuenta los diferentes problemas relacionados con las aplicaciones empresariales y los entornos de la nube, cada tipo de nube tiene sus propios beneficios e inconvenientes. Por lo tanto, la selección de un modelo de nube adecuado depende del escenario comercial específico.

Ventajas económicas: El modelo de computación en la nube es atractivo ya que libera al dueño de la empresa de la necesidad de invertir en infraestructura, alquilando los recursos de acuerdo a las necesidades y pagando solo por el uso. Además, permite disminuir los costos operativos, ya que los proveedores de servicios no tienen que provisionar capacidades de acuerdo con los picos de carga (de hecho, los recursos se liberan cuando la demanda del servicio es baja). Finalmente, la externalización de la infraestructura de servicios a las Nubes desplaza el riesgo empresarial hacia el proveedor de la infraestructura, generalmente mejor equipado para gestionarlo.

Ventajas técnicas: La computación en la nube garantiza una serie de beneficios técnicos, que incluyen: eficiencia energética, optimización de la utilización de recursos de hardware y software, elasticidad, aislamiento del rendimiento y flexibilidad; y sostenibilidad ambiental.

1.9 *Green IT*

La computación sustentable estudia el proceso mediante el cual los profesionales del ramo diseñan computadoras y subsistemas asociados, de manera eficiente y efectiva con un impacto mínimo en el medio ambiente.

El desarrollo de *Green IT* se refiere a tecnologías y sistemas de información para reducir el consumo de energía; optimizar la eficiencia energética de la infraestructura técnica; reducir los gases de efecto

invernadero inducidos por las emisiones de las tecnologías; y reducir la huella ambiental total de una empresa.

Desde esta perspectiva, puede decirse que refleja la medida en que las organizaciones adquieren y construyen una infraestructura de TIC más eficaz para el medio ambiente (Marcillo Sánchez, 2022). Esto implica tecnologías y sistemas de información para reducir el consumo de energía de alimentación y refrigeración de los activos de TIC; optimizar la eficiencia energética de la infraestructura técnica TIC; reducir las emisiones de gases de efecto invernadero inducidas por las TIC; suplantar las prácticas comerciales que emiten carbono; y analizar la huella ambiental total de una empresa (Sánchez & Barrezueta, 2023).

El *Cloud computing* ofrece servicios de TI orientados a servicios públicos a usuarios de todo el mundo. Basado en un modelo de pago por uso, permite el alojamiento de aplicaciones omnipresentes de dominios comerciales, científicos y de consumo. Sin embargo, los centros de datos que alojan aplicaciones en la nube consumen enormes cantidades de energía eléctrica, lo que contribuye a altos costos operativos y huellas de carbono para el medio ambiente. Por lo tanto, se necesitan soluciones de cómputo *Green Cloud* que no solo puedan minimizar los costos operativos sino también reducir el impacto ambiental.

1.10 Conclusiones del capítulo

En este capítulo se han abordado los elementos básicos de los sistemas computacionales; partiendo desde los componentes de un sistema computacional, los tipos de sistemas que conforma en software, el rendimiento de las computadoras. También se abordaron elementos integrales del *Cloud computing* y los elementos básicos de *Green IT*, como factor esencial de cualquier sistema de cómputo para establecer patrones y políticas responsables con el medio ambiente.

CAPÍTULO II: INTERNET DE LAS COSAS

2.1. Introducción del capítulo

El paradigma de Internet de las cosas (IoT) se basa en nodos inteligentes y autoconfigurables interconectados en una infraestructura de red dinámica y global. Representa una de las tecnologías más disruptivas, que permite escenarios informáticos ubicuos y generalizados. IoT generalmente se caracteriza por cosas pequeñas del mundo real, ampliamente distribuidas, con capacidad limitada de almacenamiento y procesamiento, lo que implica preocupaciones con respecto a la confiabilidad, el rendimiento, la seguridad y la privacidad (Alguliyev et al., 2018).

El Internet de las cosas ha surgido como un conjunto de tecnologías que van desde las Redes de Sensores Inalámbricos (WSN) hasta la Identificación por Radiofrecuencia (RFID), que brindan las capacidades para detectar, actuar y comunicarse a través de Internet. IoT ofrece un mundo infinito de posibilidades de innovación y optimización, debido a la combinación de muchos sistemas y tecnologías avanzadas como Big Data, Inteligencia Artificial, redes, *cloud computing*, objetos inteligentes, robótica, *middleware*, entre otros (Nguyen et al., 2021).

La naturaleza creciente y la interconectividad de Internet hacen que la plataforma IoT sea más escalable y fácilmente adaptable. IoT ha dejado sus marcas en los campos de la producción de alimentos, la industria manufacturera, el sector financiero, el sector doméstico, la biotecnología, el turismo, y el dominio de la salud. Tiene un respaldo de innovación que continuamente fusiona más técnicas de automatización en las plataformas IoT para mejorar un alto grado de comodidad y conveniencia para los usuarios de Internet. Cada vez más organizaciones están migrando hacia IoT, lo que a su vez aumenta la cantidad de dispositivos conectados (Nauman et al., 2020).

2.2. Componentes de IoT

Sistemas de identificación por radiofrecuencia (RFID): En el escenario de IoT, los sistemas de identificación por radiofrecuencia (RFID), compuestos por uno o más lectores y varias etiquetas, juegan un papel clave. En un escenario de uso típico, los lectores activan la transmisión de la etiqueta al generar una señal adecuada, consultando la posible presencia de objetos identificados de manera única por las etiquetas (Mulloni & Donelli, 2020). Las etiquetas RFID suelen ser pasivas (no necesitan fuente de alimentación integrada), pero también hay etiquetas alimentadas por baterías.

> Las tecnologías **RFID** permiten la identificación automática de cualquier dispositivo a la que estén conectados y permiten que a los objetos se les asignen identidades digitales únicas, se integren en una red y se asocien con información y servicios digitales.

Redes de sensores inalámbricos (WSN): Otro componente clave en los entornos de IoT está representado por las Redes de Sensores Inalámbricos (WSN). Por ejemplo, pueden cooperar con los sistemas RFID para rastrear mejor el estado de las cosas, obteniendo información sobre la posición, el movimiento, la temperatura (Noori et al., 2020). Las WSN generalmente se componen de una cantidad potencialmente alta de nodos de detección, que se comunican de

manera inalámbrica con múltiples saltos. Por lo general, se emplean nodos especiales (sumideros) para recopilar resultados.

Pueden proporcionar diversos datos útiles y se utilizan en varias áreas, como atención médica, servicios gubernamentales y ambientales, defensa, seguimiento y vigilancia de objetivos militares, exploración de entornos peligrosos, detección sísmica, etc. Sin embargo, las redes de sensores deben enfrentar muchos problemas relacionados con sus comunicaciones: corto alcance de comunicación, seguridad y privacidad, confiabilidad, movilidad. Otra limitación son los recursos: consideraciones de energía, capacidad de almacenamiento, capacidades de procesamiento, disponibilidad de ancho de banda. Además, WSN tiene sus propias limitaciones de diseño y recursos que son específicas de la aplicación y del entorno y que dependen en gran medida del tamaño del entorno de monitorización. La comunidad científica abordó en profundidad varios problemas relacionados con las redes de sensores en diferentes capas, por ejemplo: eficiencia energética, confiabilidad, robustez, escalabilidad (Naeem et al., 2020).

Direccionamiento IP: Gracias a tecnologías inalámbricas como RFID y Wi-Fi, el paradigma IoT está transformando Internet en una Internet del Futuro completamente integrada. Si bien la evolución de Internet condujo a una interconexión de personas sin precedentes, la tendencia actual conduce a la interconexión de objetos para crear un entorno inteligente. En este contexto, la capacidad de identificar cosas de forma única es fundamental para el éxito de IoT, ya que permite abordar de forma única una gran cantidad de dispositivos y controlarlos a través de Internet.

La unicidad, la confiabilidad, la persistencia y la escalabilidad representan características críticas relacionadas con la creación de un esquema de direccionamiento único.

Middleware. Debido a la heterogeneidad de los objetos participantes, a sus limitadas capacidades de almacenamiento y procesamiento y a la enorme variedad de aplicaciones involucradas, juega un papel fundamental el middleware entre las cosas y la capa de aplicación, cuyo principal objetivo es la abstracción de las funcionalidades y capacidades de comunicación de los dispositivos. El middleware se puede dividir en un conjunto de capas:

a. Abstracción de objetos.
b. Gestión de servicios.
c. Composición de servicios.
d. Aplicación.

2.3. Aplicaciones del IoT

IoT está impulsado por los avances recientes de una variedad de dispositivos y tecnologías de comunicación, pero las cosas incluidas en IoT no son solo dispositivos complejos como teléfonos móviles, sino que también comprenden objetos cotidianos como alimentos, ropa, muebles, papel, puntos de referencia, monumentos, obras de arte, etc. Estos objetos, actuando como sensores o actuadores, pueden interactuar entre sí para alcanzar un objetivo común (Singh et al., 2020).

La característica clave de IoT es, sin duda, su impacto en la vida cotidiana de los usuarios potenciales. IoT tiene efectos notables tanto en el trabajo como en el hogar, donde puede

desempeñar un papel de liderazgo en el futuro próximo: vida asistida, domótica, salud electrónica, transporte inteligente. También se esperan consecuencias importantes para los negocios, por ejemplo: logística, automatización industrial, transporte de mercancías, seguridad (Khanna & Kaur, 2020). Se ha identificado que las aplicaciones de las tecnologías IoT asistidas por la nube ganan robustez, y son aplicables en: Entornos inteligentes y ciudades inteligentes; Telemedicina; Sistema de transporte inteligente; Estacionamiento inteligente; Coordinación de drones; Cuidado de la salud. Otras aplicaciones se listan en la Tabla 1 (Villamil et al., 2020):

Tabla 1. Aplicaciones del IoT.

Dispositivo	Descripción
Bioparche	El Bioparche es un parche insertado en el paciente que determina las oscilaciones en los signos vitales.
Agricultura	Se insertan robots para monitorear cultivos. Este grupo de robots armados con inteligencia artificial pueden obtener datos cuando navegan a través de los cultivos.
SWAT robot	Este robot es enviado a áreas peligrosas para monitorear los terrenos y no exponer vidas.
Monitoreo de temperatura y humedad de cultivos	Se trata de una serie de sensores encargados de monitorear la humedad y temperatura del suelo.
Conexión de electrodomésticos	Interconecta los electrodomésticos para monitorearlos y controlarlos.
Monitoreo de personas con discapacidad	Este modelo monitorea la ubicación de personas con discapacidad.
Prevención de desastres	Predice y atiende rápidamente calamidades naturales gracias a los nodos instalados en Barcos, o en el fondo marino.
Depuradora de agua	Es una depuradora que limpia el agua mediante procesos eléctricos.
Laboratorios remotos/virtuales	Puede tomar datos de simulaciones en un laboratorio virtual basado en pruebas físicas en laboratorios remotos.
Aplicación médica para seguimiento de pacientes	Monitoriza las fichas y turnos de pacientes y personal médico a través de manillas conectadas al puesto de control del hospital.
Predicción de demanda de pacientes en hospitales	Monitorea el número de pacientes en el hospital para evitar la flexibilidad del equipo médico.

La ciudad inteligente es una de las áreas de aplicación de moda de IoT que también incorpora hogares inteligentes. El hogar inteligente consta de electrodomésticos habilitados para IoT, sistema de aire acondicionado/calefacción, televisión, dispositivos de transmisión de audio/video y sistemas de seguridad que se comunican entre sí para brindar la mejor comodidad, seguridad y un menor consumo de energía. Toda esta comunicación tiene lugar a través de una unidad de control central basada en IoT a través de Internet.

Los automóviles modernos están equipados con dispositivos y sensores inteligentes que controlan la mayoría de los componentes, desde los faros del automóvil hasta el motor. El IoT está comprometido con el desarrollo de nuevos sistemas de automóviles inteligentes que incorporen comunicación inalámbrica entre automóviles para garantizar el mantenimiento predictivo con una experiencia de conducción cómoda y segura.

Una aplicación importante de dispositivos IoT es la de los *wearables*. Un *wearable* es un dispositivo que se usa en el cuerpo humano e incluye poderosos sensores que pueden recopilar y transmitir información sobre su entorno (Verma et al., 2022). Muy a menudo, se utiliza un dispositivo portátil para realizar un seguimiento de los signos vitales sobre la salud del usuario (Dian et al., 2020). Las tecnologías portátiles se clasifican en tres categorías:

- Tecnologías de salud portátiles: dispositivos portátiles que monitorean continuamente el estado de salud de un paciente o recopilan información del mundo real sobre el paciente, como frecuencia cardíaca, presión arterial, fiebre, etc.
- Tecnologías textiles usables: por ejemplo, ropa que puede cambiar su color a pedido o según la condición biológica del usuario o según las emociones del usuario.
- Electrónica de consumo portátil: por ejemplo, pulseras, cintas para la cabeza, anillos, gafas inteligentes, relojes inteligentes.

2.4. Seguridad en el ecosistema IoT

En los últimos años, las tecnologías de *cloud computing* han contribuido a dotar al IoT de la funcionalidad necesaria para analizar y procesar información y convertirla en acciones y conocimiento en tiempo real (Gope et al., 2020). Este crecimiento sin precedentes en IoT ha abierto nuevas oportunidades para la comunidad, como mecanismos para acceder y compartir información. El paradigma de los datos abiertos es el buque insignia de estas iniciativas. Sin embargo, una de las vulnerabilidades más importantes de estas iniciativas, como ha ocurrido en muchos escenarios, es la falta de seguridad.

Las arquitecturas centralizadas como la que se usa en la computación en la nube han contribuido significativamente al desarrollo de IoT. Sin embargo, en cuanto a la transparencia de datos actúan como cajas negras y los participantes de la red no tienen una visión clara de dónde y cómo se va a utilizar la información que proporcionan (Gope et al., 2020). Todavía hay una gran cantidad de desafíos de investigación y problemas abiertos que deben estudiarse para aumentar la seguridad de los datos que se manipulan en un ecosistema IoT, y este tema de investigación aún se encuentra en una etapa preliminar. Sin embargo, se han identificado algunas mejoras (Rathee et al., 2021):

- **Descentralización y escalabilidad:** el cambio de una arquitectura centralizada a una distribuida eliminar los puntos centrales de fallas y cuellos de botella. También ayudará a prevenir escenarios en los que unas pocas empresas poderosas controlen el procesamiento y almacenamiento de la información de un gran número de personas. Otros beneficios que vienen con la descentralización de la arquitectura son una mejora de la tolerancia a fallas y la escalabilidad del sistema.
- **Identidad**: Utilizando un sistema de cadena de bloques común, los participantes pueden identificar cada dispositivo. Los datos proporcionados y alimentados al sistema

son inmutables e identifican de forma única los datos reales proporcionados por un dispositivo. Además, se puede proporcionar autenticación y autorización distribuida confiable de dispositivos para aplicaciones IoT. Esto representaría una mejora en el campo de IoT y sus participantes.

- **Confiabilidad**: la información de IoT puede permanecer inmutable y distribuida a lo largo del tiempo. Los participantes del sistema son capaces de verificar la autenticidad de los datos y tienen la certeza de que no han sido manipulados. Además, la tecnología permite la trazabilidad y la responsabilidad de los datos de los sensores. La confiabilidad es el aspecto clave de la cadena de bloques para incorporar el IoT.
- **Seguridad**: la información y las comunicaciones pueden protegerse si se almacenan como transacciones de la cadena de bloques al intercambiar mensajes de dispositivos como transacciones, validadas por contratos inteligentes, asegurando así las comunicaciones entre dispositivos.
- **Implementación segura de código**: aprovechando el almacenamiento seguro e inmutable de la cadena de bloques, el código puede insertarse de manera segura en los dispositivos. Los fabricantes pueden realizar un seguimiento de los estados y las actualizaciones con la máxima confianza. Los middlewares de IoT pueden usar esta funcionalidad para actualizar de forma segura los dispositivos de IoT.

Un aspecto fundamental de la automatización mediada por la inteligencia artificial en el IoT, es el nivel de seguridad; que es discutible, ya que en principio el código base del firmware o sistema operativo de cada dispositivo conectado o no a una red puede ser vulnerado, lo que es una falla de diseño y fabricación donde el factor seguridad fue subestimado. En este sentido, el *hackear* un sistema robótico permite sustraer datos de centros de investigación e industrias, e incluso puede causar accidentes o quitar vidas (Márquez Díaz, 2019).

2.5. Conclusiones del capítulo

La rápida evolución de la miniaturización, la electrónica y las tecnologías de comunicación inalámbrica contribuyeron a los avances del IoT. Esto ha resultado en un aumento en la cantidad de dispositivos electrónicos adecuados para muchas áreas, una reducción en sus costos de producción y un cambio de paradigma del mundo real al mundo digital. Por lo tanto, la forma en que interactúa la sociedad con el entorno cambió, utilizando la tecnología actual para obtener una mejor comprensión del mundo.

El IoT está transformando y optimizando los procesos manuales para hacerlos parte de la era digital, obteniendo volúmenes de datos que aportan conocimiento a niveles inéditos. Este conocimiento está facilitando el desarrollo de aplicaciones inteligentes como la mejora de la gestión y la calidad de vida de los ciudadanos a través de la digitalización de los servicios en las ciudades. Aunque IoT puede facilitar la digitalización de la información en sí, la confiabilidad de dicha información sigue siendo un desafío clave. En posteriores capítulos se analiza la seguridad en aplicaciones IoT.

CAPÍTULO III: PRINCIPIOS DE LA INTELIGENCIA ARTIFICIAL

3.1. Introducción del capítulo

Las Tecnologías de la Información y las Comunicaciones (TIC) han seguido evolucionando a lo largo de los años, lo que ha llevado al desarrollo de la Inteligencia Artificial (IA) (Ricardo et al., 2021). La inteligencia artificial es la capacidad de las máquinas para adaptarse a nuevas situaciones, lidiar con situaciones emergentes, resolver problemas, responder preguntas, diseñar planes y realizar varias otras funciones que requieren cierto nivel de inteligencia típicamente evidente en los seres humanos (Chen et al., 2020).

La IA es la culminación de las innovaciones y los desarrollos de las computadoras, las tecnologías relacionadas con las computadoras, las máquinas y las TIC, lo que otorga a las computadoras la capacidad de realizar funciones similares a las humanas (Loncaric et al., 2021). La IA incluye ampliamente tecnologías para el razonamiento, la resolución de problemas, la planificación y el aprendizaje, entre otras.

La Inteligencia Artificial (IA) comprende al conjunto de técnicas dedicadas a la creación de hardware y software que simulan el pensamiento humano y el comportamiento de los sistemas naturales:

- Uno de sus principios es llevar a la computadora capacidades para resolver problemas para los cuales no existen algoritmos o los que existen tienen complejidad no polinomial.
- Su fin no es reemplazar al hombre, sino proveerlo de herramientas poderosas para asistirlo en su trabajo.

La IA se usa en áreas tradicionales como:

- La búsqueda, la publicidad.
- La traducción automática.
- El reconocimiento de voz.
- Identificar clientes potenciales.
- Brindar consejos de diseño para presentaciones.
- La predicción de compras de los clientes.
- El reconocimiento de imágenes.
-

3.2. Métodos de inteligencia artificial

En comparación con los métodos tradicionales, la IA ofrece ventajas para hacer frente a problemas asociados con incertidumbres y es una ayuda eficaz para resolver problemas tan complejos. Además, las soluciones basadas en IA son buenas alternativas para determinar los parámetros de diseño de ingeniería cuando no es posible realizar pruebas, lo que genera ahorros significativos en términos de tiempo humano y esfuerzo dedicado a los experimentos. La IA también puede acelerar el proceso de toma de decisiones, disminuir las tasas de error y aumentar la eficiencia computacional. Entre las diferentes técnicas de IA, el aprendizaje automático (ML), el reconocimiento de patrones (PR) y el aprendizaje profundo (DL) han adquirido recientemente una atención considerable y se están estableciendo como una nueva clase de métodos inteligentes para su uso en ingeniería estructural (Shafri & París, 2019).

3.2.1. Métodos de solución de problemas

El objetivo del método de mantenimiento y solución de problemas basado en información integrada y principios del sistema es realizar la optimización colaborativa del diagnóstico de fallas y la solución de problemas. Para lograr este objetivo, en primer lugar, los principios de

operación del sistema, el diseño de confiabilidad del sistema y los manuales técnicos de mantenimiento se analizan exhaustivamente para obtener información de baja dimensión, por ejemplo, información de prueba, información de fallas e información de mantenimiento. Luego, los tres tipos de información de baja dimensión se integran en un modelo de integración de información de alta dimensión basado en la estructura de función del sistema a través de la correlación de fallas de prueba y el mapeo de mantenimiento de fallas basado en la estructura de funciones del sistema (Claramunt, 2020).

3.2.2. Sistemas basados en el conocimiento

Los Sistemas Basados en el Conocimiento (KBS, *Knowledge-Based Systems*) son programas informáticos basados en tecnologías establecidas por la investigación de Inteligencia Artificial, que expresan algunas características del conocimiento y la experiencia humanos para realizar tareas que normalmente realizan expertos humanos. Así, un sistema basado en conocimiento tiene dos características distintivas: una base de conocimiento y un motor de inferencia (Khan et al., 2021). La primera parte, la base de conocimiento, representa hechos sobre el mundo. La segunda parte, el motor de inferencia, permite inferir nuevos conocimientos (Blondet et al., 2019).

3.2.3. Redes neuronales artificiales

Las redes neuronales artificiales (ANN, *Artificial Neural Networks*) son herramientas esenciales en el aprendizaje automático que han llamado cada vez más la atención en la neurociencia (Musleh et al., 2019). Además de ofrecer técnicas poderosas para el análisis de datos, las ANN brindan un nuevo enfoque para que los neurocientíficos construyan modelos para comportamientos complejos, actividad neuronal heterogénea y conectividad de circuitos, así como para explorar la optimización en sistemas neuronales, en formas para las que los modelos tradicionales no están diseñados (Elsheikh et al., 2019).

3.2.4. Lógica difusa

La lógica difusa es una herramienta informática blanda bien conocida que desarrolla algoritmos viables mediante la incorporación de conocimiento humano estructurado. Es un sistema lógico que presenta un modelo diseñado para modos de interpretación humana que son más inexactos que precisos (Cabrera-Llanos et al., 2019). El sistema de lógica difusa se puede aplicar para diseñar sistemas inteligentes sobre la base de información expresada en lenguaje humano. La lógica difusa es una de las formas de inteligencia artificial; sin embargo, su historia y usos son más nuevos que los sistemas expertos basados en inteligencia artificial. La lógica difusa se ocupa de problemas que tienen imprecisión, vaguedad, aproximaciones, incertidumbre o desorden cualitativo o verdad parcial (Krishnan et al., 2020).

3.2.5. Computación con palabras

La Computación con Palabras (*Computing with Words*, CWW) es considerada una buena opción para resolver problemas de toma de decisiones con información lingüística. Los problemas cuantitativos pueden ser representados de forma precisa a través de valores

numéricos, mientras que los atributos de carácter cualitativo no pueden ser representados con un modelo de preferencia preciso, ya que la información que aporta es vaga e imprecisa. Para resolver estos problemas resulta adecuado utilizar un modelado lingüístico, donde ha sido demostrado que tiene características sobresalientes para mejorar en flexibilidad y fiabilidad (Mishra et al., 2020). El esquema básico de la CWW comienza con una entrada lingüística, donde se cuenta con una fase de unificación de la información (primera fase o traslación), posteriormente se manipula (segunda fase) y se retraslada (tercera fase), obteniéndose como resultado del proceso una salida lingüística (Tamir et al., 2019).

3.2.6. Sumarización lingüística de Datos

Una manera muy utilizada para resolver los problemas de análisis e interpretación de la información es implementando funciones de descubrimiento de conocimiento a través de la Sumarización Lingüística de Datos (LDS), ya que favorece la presentación de información al usuario, y que por tanto facilita el manejo y la interpretabilidad de dicha información por parte de este. La acción de resumir consiste en reducir una cierta cantidad de información a términos breves y precisos. No sólo es posible realizar resumen lingüístico de información representada textualmente, sino que también puede hacerse de información con otras representaciones como las numéricas (Smits et al., 2017).

La LDS es una de las poderosas técnicas de descubrimiento de conocimiento descriptivo de un gran conjunto de datos, capaz de extraer conocimiento potencial, útil y abstracto de datos tanto numéricos como categóricos (Boran et al., 2016). El resumen lingüístico de un conjunto de datos es un proceso cuya complejidad depende linealmente del tamaño del conjunto de datos y exponencialmente del tamaño del vocabulario difuso (Smits et al., 2019).

La sumarización lingüística se entiende como un lenguaje natural, generalmente corto, como una frase, capaz de resumir la esencia de un conjunto de datos numéricos o no, y por lo general demasiado grande para ser comprendido por el ser humano.

3.2.7. Teoría de los Conjuntos aproximados

Una de las teorías más recientes empleada para el análisis de datos es la Teoría de los Conjuntos Aproximados (RST). Esta teoría se considera como una de las cinco áreas claves y no tradicionales de la Inteligencia Artificial, pues constituye una herramienta muy útil para el manejo de la información no completa o imprecisa (Rahman et al., 2022). La RST Se basa en aproximar cualquier concepto, un subconjunto duro del dominio, como por ejemplo, una clase en un problema de clasificación supervisada, por un par de conjuntos exactos, llamados aproximación inferior y aproximación superior del concepto (Kida & Kida, 2022).

Con esta teoría es posible tratar tanto datos cuantitativos como cualitativos, y no se requiere eliminar las inconsistencias previas al análisis; respecto a la información de salida puede ser usada para determinar la relevancia de los atributos, generar las relaciones entre ellos (en forma de reglas), entre otras.

La teoría de los conjuntos aproximados es, potencialmente, una importante herramienta en el análisis de datos con significativas aplicaciones en la minería de datos y el descubrimiento de

conocimiento. El uso de la teoría de los conjuntos aproximados para el descubrimiento de conocimiento y minar reglas es ampliamente reconocido.

3.2.8. Metaheurísticas

Las metaheurísticas son técnicas de optimización y resolución de problemas computacionales que toman inicialmente una solución factible, la cual es luego mejorada usando procedimientos heurísticos conocidos, como recocido simulado, algoritmos genéticos, búsqueda tabú y redes neuronales. La búsqueda estocástica está presente en estos métodos y su importancia reside en ser una herramienta general de optimización cuyo estudio puede aportar mejoras para las metaheurísticas y desarrollar variantes de ellas (Tetzlaff et al., 2021).

Un método heurístico es un procedimiento que, teniendo conocimiento de un problema y de las técnicas aplicables, aporta soluciones o se acerca a ellas usando una cantidad de recursos razonable. Por lo general, este recurso es el tiempo de cómputo que se emplea en la búsqueda de la solución. Se trata de procedimientos inteligentes en el sentido de realizar una tarea que no es producto de un riguroso análisis formal ya conocido y aplicable al tema, que es a veces ineficiente en cuanto al tiempo de cómputo, sino de algoritmos de simulación, de búsqueda o evolutivos que utilizan y también aportan al conocimiento experto sobre la tarea.

3.2.9. Procesamiento de lenguaje natural

El Procesamiento de lenguaje natural (NPL, *Natural Language Processing*) es un campo de las ciencias de la computación, inteligencia artificial y la lingüística que estudia las interacciones entre las computadoras y el lenguaje humano, por medio del análisis sintáctico, semántico, pragmático y morfológico; se escriben reglas de reconocimiento de patrones estructurales, empleando un formalismo gramatical concreto. Estas reglas, en combinación con la información almacenada en diccionarios computacionales, definen los patrones que hay que reconocer en una letra palabra u oración. Se puede procesar el lenguaje natural por medio de reconocimiento de imágenes, texto y voz (Torres & Manjarrés-Betancur, 2020).

3.3. Computación evolutiva (CE)

Las técnicas de la Computación Evolutiva (CE), también conocidas como Algoritmos Evolutivos (AE), son métodos de búsqueda estocásticos que se inspiran en la genética y la selección natural de las especies. Utilizan la evolución artificial para resolver problemas de optimización, búsqueda, aprendizaje y simulación de sistemas dinámicos (Ansón Alcolea et al., 2021). Difieren de otras técnicas de optimización más tradicionales, en que realizan una búsqueda a partir de una población de soluciones, no a partir de un solo punto. Las técnicas más populares de la CE son:

3.3.1. Algoritmos Genéticos (AG):

Propuestos por Holland 1975, son un modelo de máquina de aprendizaje que deriva su comportamiento metafóricamente de los mecanismos de evolución naturales formulados por Darwin. En ellos se crea una población de individuos que pasan a través de un proceso de "evolución" simulada. Los Algoritmos Genéticos (AG) son métodos de búsqueda y optimización sobre un espacio de soluciones potenciales, basados en el principio de la selección

natural y la evolución de poblaciones. El espacio de soluciones potenciales o población es iterativamente refinado para optimizar la medida de puntaje de la población; esta medida se define a través de una función de puntaje (*fitness function*) de los individuos que se interpreta también como la habilidad de sobrevivir en el ambiente de dichos individuos (Lambora et al., 2019).

3.3.2. Programación Genética (PG):

Es la extensión del modelo genético de aprendizaje al espacio de programas. En él los objetos que constituyen la población no son cadenas de longitud fija que constituyen posibles soluciones, sino que son programas que cuando se ejecutan dan la solución al problema planteado (Ruiz Montezuma & Ospina Cadena, 2021), (Zhang et al., 2019).

3.3.3. Programación Evolutiva (PE):

Originalmente concebida por Lawrence J. Fogel en 1960, es una estrategia de optimización estocástica similar a los Algoritmos Genéticos, pero que basan su búsqueda más en las relaciones entre padres e hijos que en los mecanismos de emulación basados en los operadores genéticos (García Jiménez et al., 2021).

3.3.4. Algoritmos de Estimación de Distribuciones (EDA):

Inicialmente propuestos por H. Mühlenbein en 1996, estiman la distribución de probabilidad de los mejores individuos de cada población y generan los nuevos por simulación sobre las distribuciones estimadas. Surgen ante las dificultades de los Algoritmos Genéticos en el trabajo con variables con interacciones no lineales (Larranaga et al., 2003). Los EDAs son algoritmos heurísticos de optimización basados en poblaciones que evolucionan. En los EDAs la evolución de las poblaciones no se lleva a cabo por medio de los operadores de cruce y mutación; en lugar de ello la nueva población de individuos se muestrea a partir de una distribución de probabilidad, la cual es estimada de la base de datos que contiene al conjunto de individuos seleccionados de la generación anterior (Larrañaga et al., 2003).

3.4. Realidad Virtual

La tecnología de realidad virtual se define como un método mediante el cual se simula o replica tridimensionalmente un entorno, dando al usuario la sensación de estar dentro de él, controlándolo e interactuando personalmente con él. Los entornos virtuales son generados casi en su totalidad por computadoras. La tecnología tiene una amplia gama de beneficios probables en muchos aspectos de la vida, al proporcionar un modelo virtual tridimensional (3D) altamente detallado (Xiong et al., 2021). Al diseñar un nuevo producto, se puede usar el prototipo virtual en lugar de los prototipos físicos para evaluar el diseño desde diferentes aspectos. La realidad virtual depende de dos características básicas, a saber: la inmersión y la interacción (Wohlgenannt et al., 2020).

- **Inmersión**: es la sensación de estar presente en un entorno virtual. El entorno se genera mediante la síntesis de imágenes 3D, sonido y otros estímulos, que envuelven a los usuarios y les hacen sentir físicamente presentes en un mundo no físico. El grado en que el usuario cree estar presente en un entorno virtual (inmersión) es diferente en varios sistemas que van desde totalmente inmersivo hasta no inmersivo, dependiendo de las capacidades del sistema.

- **Interacción**: es el poder del usuario para modificar el entorno virtual. Esta característica es la principal diferencia entre las películas 3D y los entornos virtuales. En los sistemas de realidad virtual, el usuario puede interactuar con el mundo virtual, moverse por él, verlo desde diferentes ángulos, alcanzarlo, agarrarlo y remodelarlo. Este tipo de interacción es posible por medio de gafas de video montadas en la cabeza, ropa con cables y guantes de datos de fibra óptica. Los dispositivos de seguimiento de posición y la actualización en tiempo real de los sistemas de visualización auditiva y visual también son necesarios.

Los sistemas de realidad virtual se clasifican en tres grupos principales de no inmersivos, semi-inmersivos e inmersivos.

- **No inmersivos:** es la menos inmersiva y menos costosa de todas. Permite al usuario involucrarse en un entorno 3D simplemente incorporando un monitor de pantalla estéreo y gafas. Se pueden ejecutar en una computadora de escritorio estándar usando el mouse y el joystick.
- **Semi-inmersivos:** es un sistema en el que el usuario se para en una habitación con paredes de proyección trasera, piso de proyección hacia abajo, altavoces en diferentes ángulos, sensores de seguimiento en las paredes y dispositivos de sonido/música. Al usar gafas protectoras, el usuario ve todo en tres dimensiones. Dado que el usuario aún puede verse a sí mismo, el sistema no se considera un simulador totalmente inmersivo.
- **Inmersivos:** es una tecnología que brinda al usuario la experiencia psicofísica de estar completamente rodeado por un entorno virtual generado por computadora mediante el uso de hardware, software y dispositivos de interacción. La inmersión total es el nivel más alto de inmersión, que se produce mediante un dispositivo montado en la cabeza que muestra imágenes en tres dimensiones a través de un proceso conocido como estereoscopia. En este proceso, el usuario ve dos imágenes -una por ojo- y el cerebro las combina en una sola imagen 3D.

3.5. Realidad aumentada

La realidad mixta, como concepto, existe desde la concepción de crear ilusiones de realidades alternativas, popularizada por la ciencia ficción, al igual que con muchos otros avances tecnológicos. Debido a la llegada de la tecnología, la realidad mixta se ha utilizado para profesionales en áreas como la medicina, la aviación, el diseño, las humanidades, los idiomas y el entrenamiento militar. La Realidad Aumentada (AR, por sus siglas en inglés) se refiere a la superposición de gráficos generados por computadora sobre una escena del mundo real (Zhan et al., 2020). La AR une los mundos virtual y real y es una de las implementaciones comunes de la tecnología de realidad mixta. La eficacia de la implementación de AR en la capacitación es exitosa y también se ha demostrado que cumple con el objetivo de brindar capacitación profesional a los estudiantes a través de la capacitación práctica (Altınpulluk et al., 2020).

La Realidad Aumentada (AR) es una tecnología que aumenta nuestra percepción sensorial de la realidad a través de la superposición de una capa generada por computadora. Su uso más común es para potenciar el sentido de la vista, proporcionando al usuario información

contextual, imágenes tridimensionales o modelos que interactúan con el entorno y otros objetos reales

A diferencia de los simuladores de realidad virtual, en AR el entorno real no se suprime por completo y, de hecho, juega un papel dominante en este proceso. La realidad aumentada tiene como objetivo agregar aditivos sintéticos al mundo real (o a un video en vivo del mundo real) en lugar de involucrar a una persona en un mundo, que es completamente generado por una computadora. Es ampliamente utilizado en cirugía guiada por imágenes, donde los objetos reales y virtuales deben ser compuestos, integrados, presentados o manipulados simultáneamente en una sola escena (Xiong et al., 2021).

En el sector educativo, el uso de AR crea un entorno diseñado para incorporar completamente notas asistidas por AR de próxima generación, laboratorio virtual y aprendizaje interactivo basado en problemas con generación automatizada en tiempo real de escenarios centrados en aplicaciones (Rodríguez et al., 2021). Esto se puede llevar a cabo a través de la fusión y la tecnologización de materiales didácticos preexistentes (como libros y notas) utilizando AR y ser compatible con dispositivos móviles para aprovechar al máximo el aprendizaje más allá de las aulas. Este método se propone para dar a los estudiantes acceso a los recursos en cualquier momento y en cualquier lugar sin las restricciones espaciales y temporales del aprendizaje sincrónico (Lai & Cheong, 2022).

3.6. Aprendizaje automático

El mundo digital tiene una gran cantidad de datos, como datos de Internet de las cosas (IoT), datos de ciberseguridad, datos móviles, datos comerciales, datos de redes sociales, datos de salud, etc. Para analizar de forma inteligente estos datos y desarrollar las correspondientes aplicaciones inteligentes y automatizadas, el conocimiento de la inteligencia artificial (IA), en particular, el Aprendizaje Automático (ML, *Machine Learning*) es clave (Carleo et al., 2019). El aprendizaje automático se refiere a las técnicas de modelado estadístico que han impulsado el entusiasmo reciente en el mercado de software y servicios (Janiesch et al., 2021).

El núcleo del aprendizaje automático es el descubrimiento de conocimiento, el proceso de análisis basado en un conjunto de datos de muestreo conocido como datos de entrenamiento", que genera patrones significativos y un conocimiento estructurado. ML usa computadoras para simular el aprendizaje humano y permite que las computadoras identifiquen y adquieran conocimiento del mundo real, y mejoren el desempeño en algunas tareas basadas en este nuevo conocimiento.

Primero, el aprendizaje automático tiene que ver con los datos. La cantidad de esfuerzo y rigor que se necesita para descubrir, generar, administrar y versionar los datos es inherentemente más compleja y diferente que hacer lo mismo con el código de software. En segundo lugar, la creación para la personalización y la extensibilidad de los modelos requiere que los equipos no solo tengan habilidades de ingeniería de software, sino que casi siempre requieren un conocimiento lo suficientemente profundo de aprendizaje automático para construir, evaluar y ajustar modelos desde cero. En tercer lugar, puede ser más difícil mantener límites estrictos de módulos entre los componentes de aprendizaje automático que para los módulos de ingeniería de software (Huang et al., 2021).

En el área existen varios tipos de algoritmos de aprendizaje automático:

- Aprendizaje supervisado.
- Aprendizaje no supervisado.
- Aprendizaje semisupervisado
- Aprendizaje de refuerzo.

3.6.1. Aprendizaje supervisado

El aprendizaje supervisado ocurre cuando los algoritmos reciben datos de entrenamiento y respuestas correctas. La tarea del algoritmo ML es aprender en base a los datos de entrenamiento y aplicar el conocimiento que se obtuvo en datos reales. El aprendizaje supervisado suele ser la tarea del aprendizaje automático para aprender una función que asigna una entrada a una salida en función de pares de entrada-salida de muestra. Utiliza datos de entrenamiento etiquetados y una colección de ejemplos de entrenamiento para inferir una función (Ouali et al., 2020). El aprendizaje supervisado se lleva a cabo cuando se identifican ciertos objetivos que deben lograrse a partir de un determinado conjunto de insumos, es decir, un enfoque basado en tareas. Las tareas supervisadas más comunes son la clasificación que separa los datos y la regresión que ajusta los datos. Por ejemplo, predecir la etiqueta de clase o el sentimiento de un texto, como un tweet o una reseña de un producto, es decir, la clasificación del texto, es un ejemplo de aprendizaje supervisado (Jaiswal et al., 2020).

3.6.2. Aprendizaje no supervisado

En el aprendizaje no supervisado, los algoritmos de ML no tienen un conjunto de entrenamiento. Los algoritmos de aprendizaje no supervisados se centran principalmente en encontrar patrones ocultos en los datos. El aprendizaje no supervisado analiza conjuntos de datos no etiquetados sin necesidad de interferencia humana, es decir, un proceso basado en datos (Glielmo et al., 2021). Esto se usa ampliamente para extraer características generativas, identificar tendencias y estructuras significativas, agrupaciones en resultados y propósitos exploratorios. Las tareas de aprendizaje no supervisado más comunes son el agrupamiento, la estimación de densidad, el aprendizaje de características, la reducción de dimensionalidad, la búsqueda de reglas de asociación, la detección de anomalías, etc (Berry et al., 2019).

3.6.3. Aprendizaje semisupervisado

El aprendizaje semisupervisado ocurre cuando los algoritmos funcionan con un conjunto de entrenamiento al que le falta información y todavía necesitan aprender de él. El aprendizaje semisupervisado se puede definir como una hibridación de los métodos supervisados y no supervisados mencionados anteriormente, ya que opera tanto con datos etiquetados como sin etiquetar. Por lo tanto, se encuentra entre el aprendizaje “sin supervisión” y el aprendizaje “con supervisión”. En el mundo real, los datos etiquetados pueden ser raros en varios contextos, y los datos no etiquetados son numerosos, donde el aprendizaje semisupervisado es útil. Algunas áreas de aplicación en las que se utiliza el aprendizaje semisupervisado incluyen la traducción automática, la detección de fraudes, el etiquetado de datos y la clasificación de textos (Ouali et al., 2020).

3.6.4. Aprendizaje de refuerzo

El aprendizaje por refuerzo ocurre cuando los algoritmos aprenden en función de la retroalimentación externa proporcionada por una entidad pensante o el entorno. El aprendizaje por refuerzo es un tipo de algoritmo de aprendizaje automático que permite que los agentes de software y las máquinas evalúen automáticamente el comportamiento óptimo en un contexto o entorno particular para mejorar su eficiencia, es decir, un enfoque basado en el entorno. Este tipo de aprendizaje se basa en la recompensa o la penalización, y su objetivo final es utilizar los conocimientos obtenidos de los activistas ambientales para tomar medidas que aumenten la recompensa o minimicen el riesgo. Es una herramienta poderosa para entrenar modelos de IA que pueden ayudar a aumentar la automatización u optimizar la eficiencia operativa de sistemas sofisticados como robótica, tareas de conducción autónoma, fabricación y logística de la cadena de suministro; sin embargo, no es preferible usarlo para resolver problemas básicos o sencillos (Brunke et al., 2022).

3.6.5. Algoritmos de aprendizaje automático

ML se ha vuelto bastante popular recientemente con el aumento en la velocidad del procesador y el tamaño de la memoria. Como consecuencia, el campo ahora cuenta con una gran cantidad de algoritmos que usan análisis matemático o estadístico para aprender, sacar conclusiones o inferir datos.

Por lo tanto, para analizar de manera inteligente estos datos y desarrollar las correspondientes aplicaciones del mundo real, los algoritmos de aprendizaje automático son la clave. En el área de los algoritmos de aprendizaje automático, existen técnicas de análisis de clasificación, regresión, agrupamiento de datos, ingeniería de características y reducción de dimensionalidad, aprendizaje de reglas de asociación o aprendizaje de refuerzo para construir de manera efectiva sistemas basados en datos (Moerland et al., 2023).

Seleccionar un algoritmo de aprendizaje adecuado que sea adecuado para la aplicación de destino en un dominio particular es un desafío. La razón es que el propósito de diferentes algoritmos de aprendizaje es diferente, incluso el resultado de diferentes algoritmos de aprendizaje en una categoría similar puede variar según las características de los datos. Por lo tanto, es importante comprender los principios de varios algoritmos de aprendizaje automático y su aplicabilidad para aplicar en diversas áreas de aplicación del mundo real, como sistemas IoT, servicios de ciberseguridad, sistemas comerciales y de recomendación, ciudades inteligentes, atención médica, contexto -sistemas conscientes, agricultura sostenible y muchos más (Vite Cevallos et al., 2020).

Los algoritmos de aprendizaje automático generalmente consumen y procesan datos para aprender los patrones relacionados con las personas, los procesos comerciales, las transacciones, los eventos, etc.

En general, la eficacia y la eficiencia de una solución de aprendizaje automático dependen de la naturaleza y las características de los datos y del rendimiento de los algoritmos de aprendizaje. En el área de los algoritmos de aprendizaje automático, existen técnicas de análisis de clasificación, regresión, agrupamiento de datos, ingeniería de características y reducción de

dimensionalidad, aprendizaje de reglas de asociación o aprendizaje de refuerzo para construir de manera efectiva sistemas basados en datos.

Análisis de clasificación

La clasificación se considera un método de aprendizaje supervisado en el aprendizaje automático, y también se refiere a un problema de modelado predictivo, donde se predice una etiqueta de clase para un ejemplo dado. Matemáticamente, asigna una función (f) desde variables de entrada (X) a variables de salida (Y) como destino, etiqueta o categorías (Kadhim, 2019). Para predecir la clase de puntos de datos dados, se puede llevar a cabo en datos estructurados o no estructurados (Dargan et al., 2020), (Tigga & Garg, 2020). A continuación, se resumen los problemas comunes de clasificación.

Tabla 2. Problemas comunes de clasificación.

Problemas de clasificación	Descripción
Clasificación binaria	Se refiere a las tareas de clasificación que tienen dos etiquetas de clase como "verdadero y falso" o "sí y no". En tales tareas de clasificación binaria, una clase podría ser el estado normal, mientras que el estado anormal podría ser otra clase.
Clasificación multiclase	Tradicionalmente, se refiere a aquellas tareas de clasificación que tienen más de dos etiquetas de clase. La clasificación multiclase no tiene el principio de resultados normales y anormales, a diferencia de las tareas de clasificación binaria. En cambio, dentro de un rango de clases específicas, los ejemplos se clasifican como pertenecientes a una. Por ejemplo, puede ser una tarea de clasificación multiclase clasificar varios tipos de ataques de red en el conjunto de datos NSL-KDD, donde las categorías de ataque se clasifican en cuatro etiquetas de clase, como DoS (ataque de denegación de servicio), U2R (*User to Root Attack*), R2L (*Root to Local Attack*) y *Probing Attack*.
Clasificación de etiquetas múltiples	En el aprendizaje automático, la clasificación de etiquetas múltiples es una consideración importante cuando un ejemplo está asociado con varias clases o etiquetas. Por lo tanto, es una generalización de la clasificación multiclase, donde las clases involucradas en el problema están estructuradas jerárquicamente, y cada ejemplo puede pertenecer simultáneamente a más de una clase en cada nivel jerárquico, por ejemplo, clasificación de texto multinivel. La clasificación de múltiples etiquetas incluye algoritmos avanzados de aprendizaje automático que admiten la predicción de varias clases o etiquetas no exclusivas entre sí, a diferencia de tareas de clasificación tradicionales donde las etiquetas de clase son mutuamente excluyentes.

Se han propuesto muchos algoritmos de clasificación en la literatura sobre aprendizaje automático y ciencia de datos. A continuación, se resumen los métodos más comunes y populares que se utilizan ampliamente en diversas áreas de aplicación.

Naive Bayes (NB): El algoritmo *Naive Bayes* se basa en el teorema de Bayes con el supuesto de independencia entre cada par de características. Funciona bien y se puede utilizar tanto para categorías binarias como multiclase en muchas situaciones del mundo real, como clasificación de documentos o texto, filtrado de correo no deseado, etc. Para clasificar de manera efectiva las instancias ruidosas en los datos y construir un modelo de predicción sólido, se puede utilizar el clasificador NB. El beneficio clave es que, en comparación con enfoques más sofisticados, necesita una pequeña cantidad de datos de entrenamiento para estimar los parámetros

necesarios y rápidamente. Sin embargo, su desempeño puede afectar debido a sus fuertes suposiciones sobre la independencia de características. *Gaussian, Multinomial, Complement, Bernoulli y Categorical* son las variantes comunes del clasificador NB.

Análisis Discriminante Lineal (LDA): El Análisis Discriminante Lineal es un clasificador de límite de decisión lineal creado al ajustar densidades condicionales de clase a los datos y aplicar la regla de Bayes. Este método también se conoce como una generalización del discriminante lineal de Fisher, que proyecta un conjunto de datos dado en un espacio de menor dimensión, es decir, una reducción de la dimensionalidad que minimiza la complejidad del modelo o reduce los costos computacionales del modelo resultante. El modelo LDA estándar generalmente se adapta a cada clase con una densidad gaussiana, suponiendo que todas las clases comparten la misma matriz de covarianza. LDA está estrechamente relacionado con el análisis de varianza y el análisis de regresión, que buscan expresar una variable dependiente como una combinación lineal de otras características o medidas.

Regresión logística (LR): Otro modelo estadístico común basado en probabilidades utilizado para resolver problemas de clasificación en el aprendizaje automático es la regresión logística (LR). La regresión logística generalmente usa una función logística para estimar las probabilidades. Puede sobreajustar conjuntos de datos de alta dimensión y funciona bien cuando el conjunto de datos se puede separar linealmente. La suposición de linealidad entre las variables dependientes e independientes se considera un inconveniente importante de la regresión logística. Se puede usar tanto para problemas de clasificación como de regresión, pero se usa más comúnmente para la clasificación.

K-vecinos más cercanos (KNN): K-vecinos más cercanos implementa un aprendizaje basado en instancias o aprendizaje no generalizador. No se enfoca en construir un modelo interno general; en cambio, almacena todas las instancias correspondientes a los datos de entrenamiento en un espacio n-dimensional. KNN utiliza datos y clasifica nuevos puntos de datos en función de medidas de similitud como la función de distancia euclidiana. La clasificación se calcula a partir de un voto de mayoría simple de los k vecinos más cercanos de cada punto. Es bastante resistente a los datos de entrenamiento ruidosos y la precisión depende de la calidad de los datos. El mayor problema con KNN es elegir el número óptimo de vecinos a considerar. KNN se puede utilizar tanto para la clasificación como para la regresión (Morales España et al., 2008).

Máquina de vectores de soporte (SVM): En el aprendizaje automático, otra técnica común que se puede usar para clasificación, regresión u otras tareas es una máquina de vectores de soporte (SVM). En un espacio de dimensión alta o infinita, una máquina de vectores de soporte construye un hiperplano o un conjunto de hiperplanos. Intuitivamente, el hiperplano, que tiene la mayor distancia de los puntos de datos de entrenamiento más cercanos en cualquier clase, logra una fuerte separación ya que, en general, cuanto mayor es el margen, menor es el error de generalización del clasificador. Sin embargo, cuando el conjunto de datos contiene más ruido, como clases de destino superpuestas, SVM no funciona bien (Velásquez et al., 2010).

Árbol de decisión (DT): El árbol de decisión es un método de aprendizaje supervisado no paramétrico muy conocido. Los métodos de aprendizaje DT se utilizan tanto para tareas de clasificación como de regresión. Son efectivos en los dominios de aplicación relevantes, como

el análisis del comportamiento del usuario y el análisis de ciberseguridad. Al clasificar el árbol desde la raíz hasta algunos nodos hoja, DT clasifica las instancias. Las instancias se clasifican comprobando el atributo definido por ese nodo, comenzando en el nodo raíz del árbol y luego moviéndose hacia abajo en la rama del árbol correspondiente al valor del atributo (Timarán-Pereira et al., 2019).

Bosque aleatorio (RF): El clasificador de bosque aleatorio es bien conocido como una técnica de clasificación de conjuntos que se utiliza en el campo del aprendizaje automático y la ciencia de datos en varias áreas de aplicación. Este método utiliza conjuntos paralelos que ajustan varios clasificadores de árboles de decisión en paralelo, en diferentes submuestras de conjuntos de datos y utiliza votos por mayoría o promedios para el resultado final. Por lo tanto, minimiza el problema de sobreajuste y aumenta la precisión y el control de la predicción. Por lo tanto, el modelo de aprendizaje de RF con múltiples árboles de decisión suele ser más preciso que un modelo basado en un solo árbol de decisión. Para construir una serie de árboles de decisión con variación controlada, combina la agregación *bootstrap* y la selección aleatoria de características. Es adaptable tanto a problemas de clasificación como de regresión y se adapta bien a valores categóricos y continuos (Ramírez et al., 2021).

Adaptive Boosting (AdaBoost): *Adaptive Boosting* es un proceso de aprendizaje conjunto que emplea un enfoque iterativo para mejorar los clasificadores deficientes aprendiendo de sus errores. A diferencia del bosque aleatorio que usa ensamblaje paralelo, *Adaboost* usa ensamblaje secuencial. Crea un clasificador poderoso al combinar muchos clasificadores de bajo rendimiento para obtener un buen clasificador de alta precisión. En ese sentido, *AdaBoost* se denomina clasificador adaptativo porque mejora significativamente la eficiencia del clasificador, pero en algunos casos puede desencadenar sobreajustes. *AdaBoost* se utiliza mejor para aumentar el rendimiento de los árboles de decisión, el estimador base, en problemas de clasificación binaria; sin embargo, es sensible a datos con ruido y valores atípicos (Zhao et al., 2019).

Aumento de gradiente extremo (XGBoost): *Extreme Gradient Boosting* (XGBoost) es un algoritmo de aprendizaje de conjunto que genera un modelo final basado en una serie de modelos individuales, generalmente árboles de decisión. El gradiente se usa para minimizar la función de pérdida, de forma similar a cómo las redes neuronales usan el descenso de gradiente para optimizar los pesos. (XGBoost es una forma de aumento de gradiente que tiene en cuenta aproximaciones más detalladas al determinar el mejor modelo. Calcula gradientes de segundo orden de la función de pérdida para minimizar la pérdida y la regularización avanzada, lo que reduce el ajuste excesivo y mejora la generalización y el rendimiento del modelo. XGBoost es rápido de interpretar y puede manejar bien conjuntos de datos de gran tamaño (Ni et al., 2020).

Descenso de gradiente estocástico (SGD): El descenso de gradiente estocástico (SGD) es un método iterativo para optimizar una función objetivo con propiedades de suavidad apropiadas, donde la palabra estocástico se refiere a probabilidad aleatoria. Esto reduce la carga computacional, particularmente en problemas de optimización de alta dimensión, lo que permite iteraciones más rápidas a cambio de una tasa de convergencia más baja. Un gradiente es la pendiente de una función que calcula el grado de cambio de una variable en respuesta a los cambios de otra variable. Matemáticamente, el SGD es una función convexa cuya salida es

una derivada parcial de un conjunto de sus parámetros de entrada. En el aprendizaje automático escaso y a gran escala, SGD se ha aplicado con éxito a los problemas que se encuentran a menudo en la clasificación de textos y el procesamiento del lenguaje natural. Sin embargo, SGD es sensible al escalado de características y necesita una gama de hiperparámetros, como el parámetro de regularización y el número de iteraciones (Junca Peláez & Velasco Gregory, 2020).

Clasificación basada en reglas: El término clasificación basada en reglas se puede utilizar para hacer referencia a cualquier esquema de clasificación que utilice reglas $SI - ENTONCES$ para la predicción de clases. El árbol de decisión es uno de los algoritmos de clasificación basados en reglas más comunes entre estas técnicas porque tiene varias ventajas, como ser más fácil de interpretar; la capacidad de manejar datos de alta dimensión; sencillez y rapidez; buena precisión; y la capacidad de producir reglas para una clasificación humana clara y comprensible. Las reglas basadas en árboles de decisión también brindan una precisión significativa en un modelo de predicción para casos de prueba no vistos. Dado que las reglas son fácilmente interpretables, estos clasificadores basados en reglas se utilizan a menudo para producir modelos descriptivos que pueden describir un sistema, incluidas las entidades y sus relaciones (Duque-Méndez et al., 2020).

Análisis de regresión

El análisis de regresión incluye varios métodos de aprendizaje automático que permiten predecir una variable de resultado continua (y) en función del valor de una o más variables predictoras (x).

> La distinción más significativa entre clasificación y regresión es que la clasificación predice etiquetas de clase distintas, mientras que la regresión facilita la predicción de una cantidad continua.

A menudo se encuentran algunas superposiciones entre los dos tipos de algoritmos de aprendizaje automático. Los modelos de regresión ahora se usan ampliamente en una variedad de campos, que incluyen pronóstico o predicción financiera, estimación de costos, análisis de tendencias, marketing, estimación de series temporales, modelado de respuesta a medicamentos y muchos más. Algunos de los tipos familiares de algoritmos de regresión son la regresión lineal, polinómica, de lazo y de cresta, etc., que se explican brevemente a continuación (Baños et al., 2019).

Regresión lineal simple y múltiple: esta es una de las técnicas de modelado de ML más populares, así como una técnica de regresión muy conocida. En esta técnica, la variable dependiente es continua, las variables independientes pueden ser continuas o discretas y la forma de la línea de regresión es lineal. La regresión lineal crea una relación entre la variable dependiente (y) y una o más variables independientes (x) (también conocidas como línea de regresión) utilizando la línea recta de mejor ajuste. La regresión lineal múltiple es una extensión de la regresión lineal simple que permite que dos o más variables predictoras modelen una variable de respuesta, y, como una función lineal, mientras que la regresión lineal simple tiene solo una variable independiente (Cárdenas-Pérez & Echeverría, 2021).

Regresión polinomial: La regresión polinomial es una forma de análisis de regresión en la que la relación entre la variable independiente (x) y la variable dependiente(y) no es lineal, sino que es el grado polinómico de (n) en (x). La ecuación para la regresión polinomial también se deriva de la ecuación de regresión lineal (regresión polinomial de grado 1). Sin son bien conocidas como técnicas poderosas que normalmente se utilizan para construir modelos de aprendizaje en presencia de una gran cantidad de características, debido a su capacidad para evitar el sobreajuste y reducir la complejidad del modelo. Por otro lado, la regresión de cresta utiliza la regularización L2, que es la "magnitud cuadrada de los coeficientes" (penalización L2). Por lo tanto, la regresión de cresta obliga a que los pesos sean pequeños, pero nunca establece el valor del coeficiente en cero y realiza una solución no dispersa (Martínez & Huertas, 2021).

Análisis de agrupamiento (*Clustering*)

El análisis agrupamiento, es una técnica de aprendizaje automático no supervisada para identificar y agrupar puntos de datos relacionados en grandes conjuntos de datos sin preocuparse por el resultado específico. Agrupa una colección de objetos de tal manera que los objetos en la misma categoría, llamados clúster, son en cierto sentido más similares entre sí que los objetos en otros grupos. A menudo se utiliza como técnica de análisis de datos para descubrir tendencias o patrones interesantes en los datos, por ejemplo, grupos de consumidores en función de su comportamiento. En una amplia gama de áreas de aplicación, como la ciberseguridad, el comercio electrónico, el procesamiento de datos móviles, el análisis de salud, el modelado de usuarios y el análisis de comportamiento, se puede utilizar la agrupación en clústeres (Contreras Contreras et al., 2022). A continuación se analizan brevemente varios tipos de métodos de agrupación:

Tabla 3. Métodos comunes de agrupación.

Métodos de agrupación	Descripción
Métodos de partición	En función de las características y similitudes de los datos, este enfoque de agrupación clasifica los datos en varios grupos o conglomerados. Los científicos o analistas de datos suelen determinar el número de clústeres de forma dinámica o estática, según la naturaleza de las aplicaciones de destino, para producir los métodos de agrupación en clústeres.
Métodos basados en la densidad	Para identificar grupos o conglomerados distintos, utiliza el concepto de que un conglomerado en el espacio de datos es una región contigua de alta densidad de puntos aislada de otros conglomerados por regiones contiguas de baja densidad de puntos. Los puntos que no forman parte de un clúster se consideran ruido. Los métodos basados en la densidad suelen tener problemas con los grupos de densidad similar y datos de alta dimensionalidad.
Métodos basados en jerarquías	El agrupamiento jerárquico generalmente busca construir una jerarquía de agrupaciones, es decir, la estructura de árbol. Las estrategias para el agrupamiento jerárquico generalmente se dividen en dos tipos: **Aglomerativo**: un enfoque "de abajo hacia arriba" en el que cada observación comienza en su grupo y los pares de grupos se combinan como uno solo, asciende en la jerarquía. **Divisivo**: un enfoque "de arriba hacia abajo" en el que todas las observaciones comienzan en un grupo y las divisiones se realizan de forma recursiva, se mueve hacia abajo en la jerarquía.
Métodos basados en cuadrículas	Para manejar conjuntos de datos masivos, la agrupación en clústeres basada en cuadrículas es especialmente adecuada. Para obtener conglomerados, el principio es primero resumir

	el conjunto de datos con una representación de cuadrícula y luego combinar celdas de cuadrícula.
Métodos basados en modelos	Existen principalmente dos tipos de algoritmos de agrupamiento basados en modelos: uno que utiliza aprendizaje estadístico y otro basado en un método de aprendizaje de redes neuronales.
Métodos basados en restricciones	El agrupamiento basado en restricciones es un enfoque semisupervisado para el agrupamiento de datos que utiliza restricciones para incorporar el conocimiento del dominio. Se incorporan restricciones orientadas a la aplicación o al usuario para realizar el agrupamiento.

Se han propuesto muchos algoritmos de agrupamiento con la capacidad de agrupar datos en la literatura sobre aprendizaje automático y ciencia de datos. A continuación, se resumen los métodos populares que se utilizan ampliamente en diversas áreas de aplicación.

Agrupamiento de K-means: el agrupamiento de K-means es un algoritmo rápido, robusto y simple que brinda resultados confiables cuando los conjuntos de datos están bien separados entre sí. Los puntos de datos se asignan a un grupo en este algoritmo de tal manera que la cantidad de la distancia al cuadrado entre los puntos de datos y el centroide sea lo más pequeña posible. En otras palabras, el algoritmo de K-medias identifica el número k de centroides y luego asigna cada punto de datos al grupo más cercano mientras mantiene los centroides lo más pequeños posible. Dado que comienza con una selección aleatoria de centros de conglomerados, los resultados pueden ser inconsistentes. Dado que los valores extremos pueden afectar fácilmente a una media, el algoritmo de agrupación en clústeres de K-medias es sensible a los valores atípicos (Franco-Árcega et al., 2021). El agrupamiento de K-medoides es una variante de K-means que es más resistente a ruidos y valores atípicos.

Agrupamiento de desplazamiento medio: el agrupamiento de desplazamiento medio es una técnica de agrupamiento no paramétrico que no requiere conocimiento previo del número de clústeres o restricciones en la forma del clúster. El agrupamiento de cambio medio tiene como objetivo descubrir "manchas" en una distribución suave o densidad de muestras. Es un algoritmo basado en el centroide que funciona actualizando los candidatos centroides para que sean la media de los puntos en una región dada. Para formar el conjunto final de centroides, estos candidatos se filtran en una etapa de procesamiento posterior para eliminar los casi duplicados. El análisis de conglomerados en la visión artificial y el procesamiento de imágenes son ejemplos de dominios de aplicación. Tiene la desventaja de ser computacionalmente costoso. Además, en casos de gran dimensión, donde el número de conglomerados cambia abruptamente, el algoritmo de cambio medio no funciona bien (Suárez, 2020).

DBSCAN: el Agrupamiento Espacial de Aplicaciones con Ruido Basado en la Densidad (DBSCAN) es un algoritmo base para el agrupamiento basado en la densidad que se usa ampliamente en la minería de datos y el aprendizaje automático. Esto se conoce como una técnica de agrupación en clústeres basada en la densidad no paramétrica para separar los clústeres de alta densidad de los clústeres de baja densidad que se utilizan en la creación de modelos. La idea principal de DBSCAN es que un punto pertenece a un grupo si está cerca de muchos puntos de ese grupo. Puede encontrar grupos de varias formas y tamaños en un gran volumen de datos que es ruidoso y contiene valores atípicos. DBSCAN, a diferencia de K-

Means, no requiere una especificación a priori del número de grupos en los datos y puede encontrar grupos con formas arbitrarias. Aunque K-Means es mucho más rápido que DBSCAN, es eficiente para encontrar regiones de alta densidad y valores atípicos, es decir, es resistente a los valores atípicos.

Agrupamiento de GMM: los Modelos de Mezcla Gaussiana (GMM) se utilizan a menudo para el agrupamiento de datos, que es un algoritmo de agrupamiento basado en la distribución. Un modelo de mezcla gaussiana es un modelo probabilístico en el que todos los puntos de datos son producidos por una mezcla de un número finito de distribuciones gaussianas con parámetros desconocidos. Para encontrar los parámetros gaussianos para cada grupo, se puede usar un algoritmo de optimización llamado Maximización de Expectativas (EM). EM es un método iterativo que utiliza un modelo estadístico para estimar los parámetros. A diferencia de las k-medias, los modelos de mezcla gaussiana tienen en cuenta la incertidumbre y devuelven la probabilidad de que un punto de datos pertenezca a uno de los k grupos. El agrupamiento de GMM es más sólido que K-Means y funciona bien incluso con distribuciones de datos no lineales (Rendón-Hurtado et al., 2020).

Agrupación jerárquica aglomerativa: el método más común de agrupación jerárquica utilizado para agrupar objetos en clústeres en función de su similitud es la agrupación aglomerativa. Esta técnica utiliza un enfoque de abajo hacia arriba, donde cada objeto es tratado primero como un grupo único por el algoritmo. Después de eso, los pares de clústeres se fusionan uno por uno hasta que todos los clústeres se hayan fusionado en un solo clúster grande que contiene todos los objetos. El resultado es un dendograma, que es una representación de los elementos basada en un árbol. La principal ventaja del agrupamiento jerárquico aglomerativo sobre K-Means es que la jerarquía de estructura de árbol generada por el agrupamiento aglomerativo es más informativa que la colección no estructurada de clusters planos devueltos por K-Means, lo que puede ayudar a tomar mejores decisiones en las áreas de aplicación relevantes (Caroca et al., 2022). .

Aprendizaje por refuerzo

El aprendizaje por refuerzo (RL) es una técnica de aprendizaje automático que permite que un agente aprenda por ensayo y error en un entorno interactivo utilizando información de sus acciones y experiencias. A diferencia del aprendizaje supervisado, que se basa en ejemplos o datos de muestra dados, el método RL se basa en la interacción con el entorno. El problema a resolver en el aprendizaje por refuerzo (RL) se define como un Proceso de decisión de Markov (MDP), es decir, todo sobre la toma de decisiones secuencialmente. A continuación, se presentan los algoritmos RL más populares (Barsce et al., 2020).

Métodos de Monte Carlo: las técnicas de Monte Carlo, o experimentos de Monte Carlo, son una amplia categoría de algoritmos computacionales que se basan en muestreos aleatorios repetidos para obtener resultados numéricos. El concepto subyacente es utilizar la aleatoriedad para resolver problemas que son deterministas en principio. La optimización, la integración numérica y la elaboración de dibujos a partir de la distribución de probabilidad son las tres clases de problemas donde se utilizan con mayor frecuencia las técnicas de Monte Carlo (Urbina et al., 2021).

Q-learning: es un algoritmo de aprendizaje por refuerzo sin modelo para aprender la calidad de los comportamientos que le indican a un agente qué acción tomar bajo qué condiciones. No necesita un modelo del entorno, y puede lidiar con transiciones estocásticas y recompensas sin necesidad de adaptaciones. La 'Q' en Q-learning generalmente representa calidad, ya que el algoritmo calcula las recompensas máximas esperadas para un comportamiento determinado en un estado determinado (Clifton & Laber, 2020).

Deep Q-learning: el paso de trabajo básico en Deep Q-Learning es que el estado inicial se alimenta a la red neuronal, que devuelve el valor Q de todas las acciones posibles como salida. Aún así, cuando tenemos una configuración razonablemente sencilla de superar, Q-learning funciona bien. Sin embargo, cuando la cantidad de estados y acciones se vuelve más complicada, el aprendizaje profundo se puede utilizar como un aproximador de funciones (Fan et al., 2020).

Red neuronal artificial y aprendizaje profundo

El aprendizaje profundo es parte de una familia más amplia de enfoques de aprendizaje automático basados en Redes Neuronales Artificiales (ANN) con aprendizaje de representación. El aprendizaje profundo proporciona una arquitectura computacional mediante la combinación de varias capas de procesamiento, como las capas de entrada, ocultas y de salida, para aprender de los datos. La principal ventaja del aprendizaje profundo sobre los métodos tradicionales de aprendizaje automático es su mejor rendimiento en varios casos, particularmente en el aprendizaje de grandes conjuntos de datos (Díaz-Ramírez, 2021).

Los algoritmos de aprendizaje profundo más comunes son: Perceptrón multicapa (MLP), Red neuronal de red neuronal convolucional (LSTM-RNN). A continuación, se discuten varios tipos de métodos de aprendizaje profundo que se pueden usar para construir modelos efectivos basados en datos para varios propósitos:

Perceptrón Multicapa (MLP): la arquitectura base del aprendizaje profundo, que también se conoce como red neuronal artificial de avance, se denomina perceptrón multicapa (MLP). Un MLP típico es una red completamente conectada que consta de una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo de una capa se conecta a cada nodo de la siguiente capa con un cierto peso. MLP utiliza la técnica de "propagación hacia atrás", el "bloque de construcción más fundamental" en una red neuronal, para ajustar los valores de peso internamente mientras se construye el modelo. MLP es sensible a las funciones de escalado y permite ajustar una variedad de hiperparámetros, como la cantidad de capas ocultas, neuronas e iteraciones, lo que puede resultar en un modelo computacionalmente costoso (Ruelas et al., 2020).

CNN o ConvNet: La Red Neuronal de Convolución mejora el diseño de la ANN estándar, que consta de capas convolucionales, capas de agrupación y capas totalmente conectadas. Como aprovecha de la estructura bidimensional (2D) de los datos de entrada, generalmente se usa ampliamente en varias áreas, como reconocimiento de imágenes y videos, procesamiento y clasificación de imágenes, análisis de imágenes médicas, procesamiento de lenguaje natural, etc. carga, sin ninguna intervención manual, tiene la ventaja de detectar automáticamente las características importantes y, por lo tanto, la CNN se considera más poderosa que la ANN

convencional. Una Red Neuronal Convolucional es un tipo habitual de arquitectura de aprendizaje profundo, que se basa en el funcionamiento de la corteza visual cerebral. Este tipo especial de Redes Neuronales, se diferencia de las demás por el hecho de que cada una de las neuronas de las capas que la componen no recibe conexiones entrantes de todas las neuronas de la capa anterior, sino sólo de algunas de ellas. Esto simplifica enormemente el aprendizaje de la red, conllevando costes computacionales mucho menores que en el caso anterior (Lu et al., 2021).

LSTM-RNN: La memoria a corto plazo (LSTM) es una arquitectura de Red Neuronal Recurrente Artificial (RNN) utilizada en el área del aprendizaje profundo. LSTM tiene enlaces de retroalimentación, a diferencia de las redes neuronales de avance normales. Las redes LSTM son adecuadas para analizar y aprender datos secuenciales, como clasificar, procesar y predecir datos basados en datos de series temporales, lo que las diferencia de otras redes convencionales. Por lo tanto, LSTM se puede usar cuando los datos están en un formato secuencial, como tiempo, oración, etc., y se aplica comúnmente en el área de análisis de series de tiempo, procesamiento de lenguaje natural, reconocimiento de voz, etc.

3.7. Aplicaciones de la IA y el aprendizaje automático

En la Tabla 4 se resumen algunas áreas de aplicación populares de la IA y el aprendizaje automático (Ospina-Gutiérrez & Aristizábal, 2021), (Vásquez-Quispesivana et al., 2022), (Méndez-Gurrola, 2020).

Tabla 4. Áreas de aplicación populares de la IA y el aprendizaje automático.

Análisis predictivo y toma de decisiones inteligente	La base del análisis predictivo es capturar y explotar las relaciones entre las variables explicativas y las variables pronosticadas de eventos anteriores para predecir el resultado desconocido. Por ejemplo, la identificación de sospechosos o delincuentes después de que se haya cometido un delito, o la detección de fraudes con tarjetas de crédito a medida que ocurren. Varios algoritmos de aprendizaje automático como árboles de decisión, máquinas de vectores de soporte, redes neuronales artificiales, etc., se usan comúnmente en el área. Dado que las predicciones precisas brindan información sobre lo desconocido, pueden mejorar las decisiones de las industrias, los negocios y casi cualquier organización, incluidas las agencias gubernamentales, el comercio electrónico, las telecomunicaciones, los servicios bancarios y financieros, la atención médica, las ventas y el marketing, el transporte, las redes sociales, y muchos otros.
Clasificación de objetos y conducción autónoma	La clasificación de objetos es una tarea de investigación importante en el dominio de la inteligencia artificial (IA) y la visión por computadora y se ha aplicado a la comprensión de la escena, los robots domésticos de servicio y los sistemas de fábricas inteligentes. Para la conducción autónoma, los resultados de clasificación de objetos de alta precisión permiten que un sistema de conducción inteligente identifique obstáculos y logre una planificación de ruta autónoma segura. Los resultados de clasificación de objetos de alta precisión son vitales como paso preliminar para el trabajo posterior. Los sensores de detección y distancia de luz LiDAR se han empleado cada vez más para la clasificación de objetos porque estos sensores pueden obtener una gran cantidad de nubes de puntos 3D precisas y de alta resolución del entorno circundante.
Ciberseguridad e inteligencia de amenazas:	La ciberseguridad es una de las áreas más esenciales de los sistemas computacionales, que suele ser la práctica de proteger redes, sistemas, hardware y datos de ataques digitales. El aprendizaje automático se ha convertido en una tecnología crucial de ciberseguridad que aprende constantemente mediante el análisis de datos para identificar patrones, detectar

mejor el malware en el tráfico cifrado, encontrar amenazas internas, predecir dónde están los vecindarios malos en línea, mantener a las personas seguras mientras navegan o proteger los datos en la nube al descubrir actividades sospechosas. Por ejemplo, las técnicas de agrupamiento se pueden usar para identificar anomalías cibernéticas, violaciones de políticas, etc. Para detectar varios tipos de ataques cibernéticos o intrusiones, son útiles los modelos de clasificación de aprendizaje automático teniendo en cuenta el impacto de las características de seguridad.

IoT y ciudades inteligentes: La ciudad inteligente es uno de los principales campos de aplicación de IoT y utiliza tecnologías para mejorar los servicios de la ciudad y las experiencias de vida de los residentes. Dado que el aprendizaje automático utiliza la experiencia para reconocer tendencias y crear modelos que ayuden a predecir comportamientos y eventos futuros, se ha convertido en una tecnología crucial para las aplicaciones de IoT. Por ejemplo, predecir el tráfico en ciudades inteligentes, predecir la disponibilidad de estacionamiento, estimar el uso total de energía de los ciudadanos durante un período en particular, tomar decisiones contextuales y oportunas para las personas, etc. son algunas tareas que pueden resolverse utilizando técnicas de aprendizaje automático de acuerdo a las necesidades actuales de las personas.

Predicción del tráfico y transporte: Varias ciudades alrededor del mundo están experimentando un aumento excesivo en el volumen de tráfico, lo que resulta en serios problemas como demoras, congestión del tráfico, precios más altos del combustible, aumento de la contaminación por CO2, accidentes, emergencias y una disminución en la calidad de vida de la sociedad moderna. Por lo tanto, es importante un sistema de transporte inteligente a través de la predicción del tráfico futuro, que es una parte indispensable de una ciudad inteligente. La predicción precisa del tráfico basada en modelos de aprendizaje automático y profundo puede ayudar a minimizar los problemas. Por ejemplo, según el historial de viajes y la tendencia de viajar a través de varias rutas, el aprendizaje automático puede ayudar a las empresas de transporte a predecir posibles problemas que pueden ocurrir en rutas específicas y recomendar a sus clientes que tomen un camino diferente.

Comercio electrónico y recomendaciones de productos: La recomendación de productos es una de las aplicaciones de aprendizaje automático más conocidas y utilizadas, y es una de las características más destacadas de casi cualquier sitio web de comercio electrónico en la actualidad. La tecnología de aprendizaje automático puede ayudar a las empresas a analizar los historiales de compra de sus consumidores y hacer sugerencias de productos personalizados para su próxima compra en función de su comportamiento y preferencias. El futuro de las ventas y el marketing es la capacidad de capturar, evaluar y utilizar los datos de los consumidores para brindar una experiencia de compra personalizada. Además, las técnicas de aprendizaje automático permiten a las empresas crear paquetes y contenido que se adaptan a las necesidades de sus clientes, lo que les permite mantener a los clientes existentes y atraer a otros nuevos.

Procesamiento del lenguaje natural (PNL) y análisis de sentimientos El procesamiento del lenguaje natural (PNL) implica la lectura y comprensión del lenguaje hablado o escrito a través de una computadora. Muchas divisiones de la empresa trabajan en el procesamiento del lenguaje natural, desarrollando herramientas para el reconocimiento de entidades, análisis de sentimientos, predicción de intenciones, resúmenes, traducción automática, construcción de ontologías, similitud de texto y conexión de respuestas a preguntas. Por lo tanto, NLP ayuda a las computadoras, por ejemplo, a leer un texto, escuchar un discurso, interpretarlo, analizar sentimientos y decidir qué aspectos son significativos, donde se pueden usar técnicas de aprendizaje automático. Asistente personal virtual, chatbot, reconocimiento de voz, descripción de documentos, idioma o traducción automática, etc. son algunos ejemplos de tareas relacionadas con la PNL.

3.8. Conclusiones del capítulo

La inteligencia artificial (IA), en particular, el aprendizaje automático (ML) ha crecido rápidamente en los últimos años en el contexto del análisis de datos y la computación que, por lo general, permite que las aplicaciones funcionen de manera inteligente. ML generalmente brinda sistemas con la capacidad de aprender y mejorar a partir de la experiencia automáticamente sin ser programado específicamente. En general, la eficacia y la eficiencia de una solución de aprendizaje automático dependen de la naturaleza y las características de los datos y del rendimiento de los algoritmos de aprendizaje

El aprendizaje por refuerzo, junto con el aprendizaje supervisado y no supervisado, es uno de los paradigmas básicos del aprendizaje automático. RL se puede utilizar para resolver numerosos problemas del mundo real en varios campos, como teoría de juegos, teoría de control, análisis de operaciones, teoría de la información, optimización basada en simulación, fabricación, logística de la cadena de suministro, sistemas multiagente, inteligencia de enjambre, control de aeronaves, robot control de movimiento, y muchos más.

Las redes neuronales son el método más extendido actualmente para aplicar aprendizaje automático. Una neurona artificial intenta imitar el comportamiento que tiene una neurona biológica, imitando cada una de sus partes. Así, la información que llega a las neuronas biológicas a través de las dendritas de otras neuronas son los datos de entrada de una neurona artificial y las conexiones sinápticas se traducen a pesos sinápticos computacionales.

En general, con base en las técnicas de aprendizaje discutidas anteriormente, se puede concluir que varios tipos de aprendizaje automático, agrupamiento de datos, selección y extracción de características y reducción de dimensionalidad, aprendizaje de reglas de asociación, aprendizaje de refuerzo o técnicas de aprendizaje profundo pueden desempeñar un papel importante para varios propósitos de acuerdo a sus capacidades.

CAPÍTULO IV: INDUSTRIA 4.0

4.1. Introducción del capítulo

En 2011, el gobierno alemán trajo al mundo un nuevo rumbo llamado Cuarta Revolución Industrial (I4.0 o Industra 4.0), asumida como la cuarta revolución industrial. El objetivo de I4.0 es trabajar con un mayor nivel de automatización logrando un mayor nivel de productividad y eficiencia operativa. Industria 4.0 conduce a la era de la digitalización. Todo es digital; modelos de negocio, entornos, sistemas de producción, máquinas, operadores, productos y servicios (Carrillo et al., 2020).

Todo está interconectado dentro de la escena digital con la representación virtual correspondiente. Trae un buen desarrollo para el escenario industrial centrándose en la creación de productos inteligentes, procesos inteligentes y procedimientos inteligentes. Las empresas esperaban aumentar el nivel de digitalización, trabajando juntos en ecosistemas digitales con clientes y proveedores (Chacón-Ramírez et al., 2020).

La I4.0 está basada en la integración de datos y conocimientos heterogéneos y puede resumirse como un proceso de fabricación interoperable, integrada, adaptada, optimizada, orientada a servicios que se correlaciona con algoritmos, Big Data (BD) y altas tecnologías como Internet de las Cosas (IoT) y Servicios (IoS), Automatización Industrial, Ciberseguridad (CS), Computación en la Nube (CC) o Robótica Inteligente (Amaya et al., 2023).

4.2. Las tecnologías clave de I4.0

I4.0 se caracteriza en manufactura y servicios por procesos de automatización y digitalización, electrónica e informática altamente desarrollados. Desde la perspectiva de la gestión de la producción y el servicio, I4.0 se centra en establecer sistemas inteligentes y comunicativos, como la interacción máquina-máquina y hombre-máquina, que se ocupan del flujo de datos de la interacción del sistema inteligente y distribuido. Entre otras características, I4.0 promueve la interoperabilidad autónoma, la agilidad, la flexibilidad, la toma de decisiones, la eficiencia o la reducción de costes (R. P. García et al., 2020).

Internet industrial de las cosas: El Internet industrial de las cosas (IIoT) es la conexión de productos industriales como componentes y/o máquinas a Internet. Por ejemplo, al vincular los datos de detección recopilados en una fábrica con la plataforma IoT, IIoT aumenta la eficiencia de producción con el análisis BD (Valencia & Portilla, 2019).

Internet de los Servicios: Reemplazando las cosas físicas por servicios, Internet of Services (IoS) se basa en el concepto de que los servicios están disponibles a través de Internet para que los usuarios privados y/o las empresas puedan crear, combinar y ofrecer nuevos tipos de servicios de valor agregado (Cardenas, 2019). IoS puede permitir que los proveedores de servicios ofrezcan sus servicios en Internet. Por lo tanto, la tendencia de la industria manufacturera orientada al producto está cambiando rápidamente hacia la orientada al servicio para permitir obtener ingresos a lo largo de todo el ciclo de vida de un sistema de servicio del producto. Con esto, puede permitir productos de alta calidad y, en paralelo, brinda una posición competitiva sólida para las empresas a través de los servicios de valor agregado. IoS permite

recopilar información del producto, por ejemplo, durante su operación, para actualizaciones y para el desarrollo de nuevos servicios, aumentando la calidad percibida del producto.

Computación en la nube: *Cloud Computing* (CC) es una tecnología alternativa para las empresas que intentan invertir en recursos de externalización de TI. La adopción de CC tiene varias ventajas relacionadas con la reducción de costos, por ejemplo, los costos directos e indirectos en la eliminación de la infraestructura de TI en la organización, el servicio de racionalización de recursos por parte de los usuarios dinámicamente escalables que consumen solo los recursos informáticos que realmente usan o la portabilidad al usar cualquier tipo de dispositivo conectado a internet como teléfonos móviles o tabletas accediendo desde cualquier lugar del mundo. Todo se trata como un servicio en CC. Estos servicios definen un sistema en capas o tipos de modelos de servicio estructurados para CC (Quezada-Sarmiento & Suárez Guerrero, 2021):

- **Infraestructura como servicio (IaaS)**: es donde los proveedores de servicios en la nube brindan a los usuarios recursos informáticos fundamentales, con infraestructuras virtuales, por ejemplo, servidores virtuales, redes o almacenamiento, y donde los usuarios en la nube pueden implementar y ejecutar software arbitrario, que puede incluir, por ejemplo , aplicaciones de sistemas operativos;
- **Plataforma como servicio (PaaS):** es donde los usuarios desarrollan y ejecutan aplicaciones utilizando lenguajes de programación en las infraestructuras de la nube. Por lo tanto, se puede lograr escalabilidad, servidor y almacenamiento de alta velocidad. Los usuarios pueden crear, ejecutar e implementar sus propias aplicaciones con el uso de plataformas de TI remotas. En esta capa, no hay preocupación por la disponibilidad y el mantenimiento del recurso.
- **El software como servicio (SaaS):** es donde las aplicaciones residen y se ejecutan en una infraestructura de nube. Accesible desde varios dispositivos cliente a través de una interfaz como un navegador web y programas. El enfoque es eliminar las aplicaciones de servicio en dispositivos locales de usuarios individuales, logrando una alta eficiencia y rendimiento para los usuarios. Esta categoría permite aplicaciones de software como el software de Diseño Asistido por Computadora (CAD) y el software de Planificación de Recursos Empresariales (ERP), con un costo total de propiedad más bajo.

Cloud Manufacturing (CMfg): Con la combinación de tecnologías avanzadas, surge un nuevo modo de computación y fabricación orientado a servicios como CMfg. Una solución como CMfg permite a los usuarios solicitar servicios de todas las etapas del ciclo de vida de un producto, desde el diseño, la fabricación, la gestión. En este sentido, las principales características de CMfg son el enfoque orientado al servicio y su tendencia a cambiar el enfoque de fabricación de orientado a la producción a orientado al servicio (Wan et al., 2020).

En entornos de fabricación, se propuso el concepto *Cloud Manufacturing (CMfg)* para hacer uso de la tecnología CC, con el fin de mejorar los sistemas de fabricación actuales. Los sistemas CMfg como un tipo completamente nuevo de servicio en la nube, basado en la arquitectura orientada a servicios (SoA) en el entorno de la nube que proporciona capacidades de fabricación.

El modelo CMfg, que consta de tres categorías de partes interesadas: proveedores, operadores y consumidores, con su cooperación para mantener el funcionamiento sostenible de un sistema CMfg:

Tabla 5. Categorías de *stakeholders* en el modelo CMfg.

Categoría	Descripción
Proveedores	Poseen y proporcionan las habilidades y los recursos de fabricación.Dentro de todo el ciclo de vida del producto, con el fin de compartir, los proveedores publican recursos de fabricación en la plataforma CMfg y también reciben tareas de fabricación desde la plataforma en la nube. Todo se transforma en servicios, bajo la gestión exclusiva del operador.
Operadores	Para operar la plataforma CMfg y prestar servicios a proveedores, consumidores e incluso a terceros. Bajo demanda, los consumidores de la plataforma en la nube pueden obtener servicios de fabricación sostenibles y de alta calidad. Los proveedores tienen permiso para publicar sus recursos y capacidades con el uso de herramientas proporcionadas por la plataforma en la nube.
Consumidores	Para suscribir la disponibilidad de servicios informáticos de fabricación en una plataforma de servicios CMfg. Bajo la gestión exclusiva del operador, los consumidores, incluidos los consumidores empresariales y los consumidores individuales, envían sus tareas de requisitos a la plataforma CMfg, por ejemplo, tareas de diseño, fabricación, prueba o simulación, y también reciben los resultados de ejecución de sus pedidos.

CMfg es un paradigma de fabricación basado en el Conocimiento. En el proceso en ejecución, el conocimiento juega un papel central, por ejemplo, modelos, estándares, protocolos, reglas y algoritmos como conocimiento, indispensable en muchos procesos y actividades dentro de todo el ciclo de vida de los servicios como generación de servicios, gestión de servicios y aplicaciones de servicios.

Simulación: Los desafíos actuales de la industria manufacturera pueden ser abordados por esta tecnología, lidiando con la complejidad de los sistemas, con elementos de problemas inciertos que no pueden resolverse con los modelos matemáticos habituales. En un entorno de fabricación de productos personalizados, el valor de la simulación es notable y evidente. Para la implementación exitosa de la fabricación digital, una herramienta indispensable y poderosa, la simulación por computadora, se está convirtiendo en una tecnología para comprender mejor la dinámica de los sistemas comerciales. La simulación se define como una imitación de operaciones, en el tiempo, de un sistema o proceso del mundo real. Utiliza la historia artificial de un sistema y su observación, extrayendo inferencias sobre las características operativas de la representación del sistema real (García-García et al., 2020).

La simulación permite realizar experimentos para la validación de productos, procesos o diseño y configuración de sistemas. El modelado de simulación ayuda a reducir costos, disminuir los ciclos de desarrollo y aumentar la calidad del producto. Con el fin de analizar sus operaciones y apoyar la toma de decisiones, los fabricantes han estado utilizando modelos y simulaciones. Las tecnologías de simulación ya han demostrado su eficacia en el enfoque de varios problemas prácticos del mundo real en el sector manufacturero.

Una simulación de alta fidelidad de una fábrica de fabricación se define como Virtual Factory (VF). Un entorno de colaboración industrial centrado en la representación de Realidad Virtual (VR) de una fábrica o una instalación de emulación puede considerarse un VF. La visión VF considera modelos de simulación de fábricas reales validados para generar datos y trabajar en formatos de condiciones reales en una fábrica real.

La tecnología de simulación sobre I4.0, utilizando VR, es un proceso integral para simular todos los procesos industriales, desde la planificación, diseño, fabricación, prestación de servicios, mantenimiento, pruebas o incluso controles de calidad. Todos los procesos se pueden simular como modulares. Es posible simular y verificar virtualmente un proceso de fabricación en fábrica antes de que se realice. Después de la aprobación, se pueden hacer todos los exámenes físicos. La producción de prototipos permite la reducción de tiempo en el proceso de diseño y producción, al reducir las dependencias de valor agregado. Estas reducciones de tiempo son particularmente relevantes en mercados personalizados.

Realidad aumentada: La tecnología AR se puede encontrar en una amplia gama de sectores, por ejemplo, entretenimiento, marketing, turismo, cirugía, logística, fabricación, mantenimiento. Como una tecnología en evolución creciente, recientemente, el uso de AR se está extendiendo a diferentes campos de fabricación. Se ha demostrado que el uso de AR en los procesos de fabricación con respecto a la simulación, la asistencia y la orientación es una tecnología eficiente que ayuda en los problemas. La tecnología AR aumenta la percepción del operador de la realidad al hacer uso de información artificial sobre el entorno, donde el mundo real se cumple con sus objetos. Siempre que interactúe con los sentidos humanos, AR puede hacer uso de cualquier tipo de hardware (G. G. García et al., 2020).

Big Data (BD): Gran cantidad de datos generados de diferentes tipos, pueden provenir de objetos heterogéneos interconectados. Esta enorme cantidad de datos estructurados, semiestructurados y no estructurados puede describir Big Data (BD). Para obtener el valor correspondiente, estos datos requerirían demasiado tiempo y dinero para ser almacenados y analizados. Brindar oportunidades de valor a las industrias en la era de Internet se puede lograr con la conexión de más dispositivos físicos a Internet y con el uso de una generación de tecnologías novedosas (Nguyen et al., 2020).

La recopilación o el almacenamiento de datos caracterizan a BD, pero la característica central de BD es el análisis de datos y, sin él, BD no tiene mucho valor. BD puede proporcionar una guía sistemática para las actividades de producción relacionadas dentro del ciclo de vida completo del producto, logrando un funcionamiento rentable del proceso y sin fallas, y ayudar a los gerentes en la toma de decisiones y/o para resolver problemas relacionados con operación. El uso de BD proporciona una ventaja comercial a través de la oportunidad de generar valor agregado.

4.3. Seguridad de la información en la Industria 4.0

La seguridad de la información y los datos son fundamentales para el éxito de la industria. Es importante que los datos estén disponibles solo para las personas autorizadas. Se debe verificar la integridad y las fuentes de información. I4.0 ha planteado dos demandas de seguridad para asegurar los sistemas de fabricación inteligente: arquitectura de seguridad y seguridad por

diseño. Por lo tanto, los sistemas deben detectar automáticamente los ataques, las amenazas y el malware sin instalación (Javaid & Haleem, 2019). Las operaciones de fabricación pueden cerrarse por un ataque cibernético, por lo tanto, las empresas tienen pérdidas de dinero, pero el problema principal son los ataques cibernéticos dirigidos a sistemas que requieren operaciones de seguridad y representan un riesgo grave para la seguridad de los operadores. Algunos ataques potenciales en la Industria 4.0 son:

- La modificación de diseños de productos, relacionados con archivos CAD, tolerancias.
- La modificación de procesos de fabricación, archivos de fabricación asistida por computadora.
- Parámetros de máquinas, herramientas utilizadas, trayectorias de herramientas.
- Manipulación de datos de proceso/producto, resultados de inspección, indicadores de mantenimiento de la máquina.

Los ataques directos a la industria pueden ser graves peligros para los Sistemas de Control Industrial (ICS). Estos SCI de los sectores industriales son básicamente de control como Supervisión, Control y Adquisición de Datos (SCADA), sistemas de control de procesos, sistemas de control distribuido, CPS o Controladores Lógicos Programables (PLC). Estos ataques pueden retrasar el lanzamiento de un producto, provocar la producción de productos modificados, arruinar la confianza del cliente o aumentar los costos de garantía.

La seguridad del ICS es sensible al tiempo, por lo que se necesita una respuesta automática a incidentes. Para una variedad de ataques industriales, las Redes Definidas por Software (SDN) y la Virtualización de Funciones de Red (NFV) pueden facilitar la respuesta automática a incidentes. La respuesta a incidentes en ICS se puede lograr utilizando una arquitectura de nube privada (inversión rentable). SDN y NFV hacen posible la respuesta automática a incidentes para detectar rápidamente y reemplazar temporalmente los sistemas defectuosos con implementaciones virtuales de esos sistemas. SDN y NFV son tecnologías para mejorar los siguientes aspectos:

1. Visibilidad de red.
2. Capacidades de red (permite flujos de tráfico de red con una mejor gestión.
3. Implementación y control de funciones de red mediante software, en lugar de middleboxes de hardware específicos.

Sin embargo, la combinación de SDN con NFV muestra un enfoque capaz en nuevas soluciones de defensa en profundidad para ICS.

4.4. Conclusiones del capítulo

Para explorar datos, se requiere un análisis de datos avanzado. Usando *cloud computing* a través de análisis, métodos y herramientas avanzados, se analizan y extraen datos fuera de línea y en tiempo real, por ejemplo, aprendizaje automático, modelos de pronóstico, entre otros. El conocimiento se extrae de la gran cantidad de datos que permiten a los fabricantes comprender las diversas etapas del ciclo de vida del producto. Además, el análisis avanzado de BD se puede utilizar como facilitador, identificando y superando los cuellos de botella creados por los datos generados por IoT.

El uso de tecnologías habilitadoras es la estrategia que está detrás del paradigma I4.0. En un contexto industrial, cada tecnología implementada de manera individual presentará un menor impacto.

CAPÍTULO V. HERRAMIENTAS DE CIENCIA DE DATOS

5.1 Introducción del capítulo

Es la era de los datos, donde todo está conectado a una fuente de datos y todo se registra digitalmente. Por ejemplo, el mundo electrónico actual tiene una gran cantidad de diversos tipos de datos, como datos de Internet de las cosas (IoT), datos de seguridad cibernética, datos de ciudades inteligentes, datos comerciales, datos de teléfonos inteligentes, datos de redes sociales, datos de salud, y muchos más.

Así como la ingeniería de software se trata principalmente del código que forma el software de envío, ML se trata de los datos que impulsan los modelos de aprendizaje. Los ingenieros de software prefieren diseñar y construir sistemas que sean elegantes, abstractos, modulares y simples. Por el contrario, los datos utilizados en el aprendizaje automático son voluminosos, específicos del contexto, heterogéneos y, a menudo, complejos de describir. Estas diferencias dan como resultado problemas difíciles cuando los modelos de Aprendizaje Automático (ML) se integran en sistemas de software a escala (Rodriguez et al., 2021).

Los ingenieros tienen que buscar, recopilar, curar, limpiar y procesar datos para usarlos en el entrenamiento y ajuste de modelos. Todos los datos deben almacenarse, rastrearse y versionarse. Los conjuntos de datos rara vez tienen definiciones de esquema explícitas para describir las columnas y caracterizar sus distribuciones estadísticas. Sin embargo, debido a la rápida iteración involucrada en ML, el esquema de datos (y los datos) cambian con frecuencia, incluso muchas veces al día. Cuando se incorporan datos de fuentes de datos de diagnóstico a gran escala, si los ingenieros de ML desean cambiar los valores de datos que se recopilan, deben esperar a que los sistemas de ingeniería se actualicen, implementen y propaguen antes de que puedan llegar nuevos datos. Incluso los cambios simples pueden tener impactos significativos en el volumen de datos recopilados, lo que puede afectar las aplicaciones a través de características de rendimiento alteradas o un mayor uso del ancho de banda de la red.

Si bien existen tecnologías muy bien diseñadas para versionar el código, no ocurre lo mismo con los datos. Un conjunto de datos dado puede contener datos de varios regímenes de esquema diferentes. Cuando un solo ingeniero recopila y procesa estos datos, puede realizar un seguimiento de estos detalles no escritos, pero cuando los tamaños de los proyectos aumentan, mantener este conocimiento tribal puede convertirse en una carga.

5.2. Tipos de datos del mundo real

Por lo general, la disponibilidad de datos se considera la clave para construir un modelo de aprendizaje automático o sistemas del mundo real basados en datos. Los datos pueden ser de varias formas, como estructurados, semiestructurados o no estructurados y los metadatos (Sandoya, 2022).

- **Estructurado**: Tiene una estructura bien definida, se ajusta a un modelo de datos siguiendo un orden estándar, altamente organizado y de fácil acceso, y utilizado por una entidad o un programa informático. En esquemas bien definidos, como las bases de datos relacionales, los datos estructurados normalmente se almacenan, es decir, en un formato

tabular. Por ejemplo, nombres, fechas, direcciones, números de tarjetas de crédito, información bursátil, geolocalización, etc. son ejemplos de datos estructurados.

- **No estructurados**: por otro lado, no existe un formato u organización predefinidos para los datos no estructurados, lo que los hace mucho más difíciles de capturar, procesar y analizar, ya que en su mayoría contienen texto y material multimedia. Por ejemplo, los datos de sensores, correos electrónicos, entradas de blogs, wikis y documentos de procesamiento de texto, archivos PDF, archivos de audio, videos, imágenes, presentaciones, páginas web y muchos otros tipos de documentos comerciales pueden considerarse datos no estructurados.
- **Semiestructurados**: los datos semiestructurados no se almacenan en una base de datos relacional como los datos estructurados mencionados anteriormente, pero tienen ciertas propiedades organizativas que facilitan su análisis.
- **Metadatos**: No es la forma normal de datos, sino "datos sobre datos". La principal diferencia entre datos y metadatos es que los datos son el material que puede clasificar, medir o incluso documentar algo relativo a las propiedades de datos de una organización. Por otro lado, los metadatos describen la información de datos relevante, dándole más importancia para los usuarios de datos.

5.3. Disponibilidad, recopilación, limpieza y gestión de datos

Dado que muchas técnicas de aprendizaje automático se centran en el aprendizaje de grandes conjuntos de datos, el éxito de los proyectos centrados en ML a menudo depende en gran medida de la disponibilidad, la calidad y la gestión de los datos. Etiquetar conjuntos de datos es costoso y requiere mucho tiempo, por lo que es importante que estén disponibles para su uso dentro de la empresa (sujeto a restricciones de cumplimiento) (Pérez Rave, 2019).

La automatización es una preocupación transversal vital, que permite a los equipos agregar datos, extraer características y sintetizar ejemplos etiquetados de manera más eficiente. La mayor eficiencia permite a los equipos acelerar la experimentación y trabajar con datos en vivo mientras experimentan con nuevos modelos. El proceso de limpieza y construcción de los repositorios se concibe alineado completamente con los procesos de minería de datos que se muestran en la Figura 2, y que se describen a continuación:

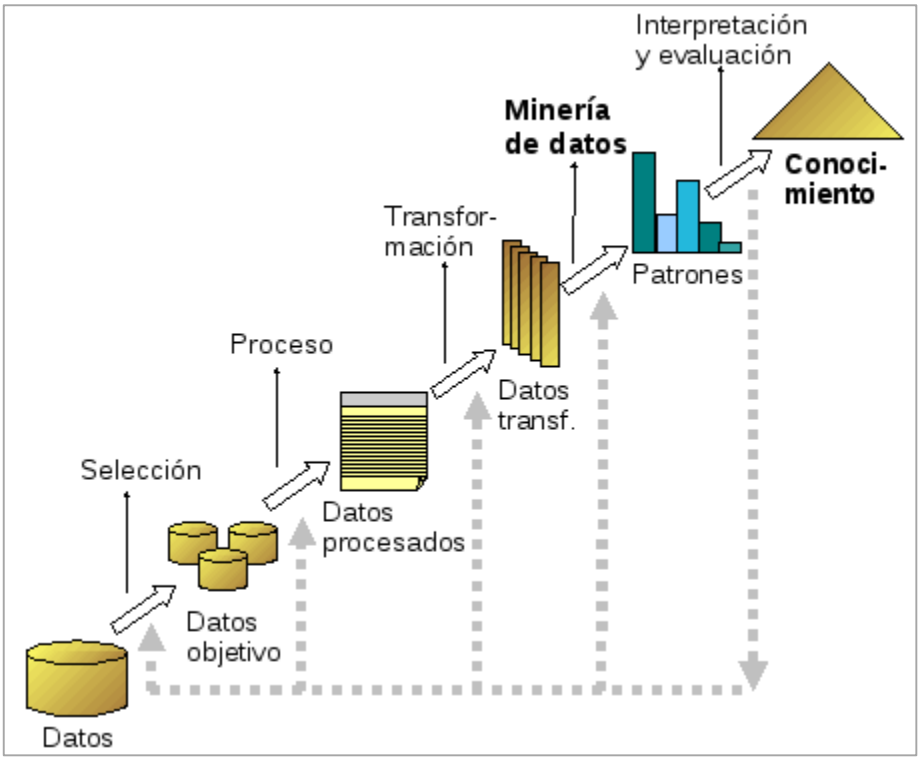


Figura 2. Procesos que intervienen en el análisis de los datos.

- **Selección:** Tiene como objetivo la preparación de las fuentes de datos y la selección de las mismas.
- **Preprocesamiento y transformación**: Se aplican técnicas como limpieza de datos, la integración y transformación de datos, la reducción de datos y la selección de atributos.
- **Aplicar las técnicas de minería**: Es el proceso del cálculo de resúmenes y valores derivados. En esta etapa se perfeccionan constantemente las técnicas y algoritmos que se encargan de extraer y representar el conocimiento de forma adecuada para la toma de decisiones. Se combinan técnicas potenciando las ventajas de cada una y atenuando sus debilidades.
- **Interpretación y evaluación:** en este paso se procede al análisis de los resultados descubiertos. Incluye a su vez la resolución de posibles inconsistencias con otros conocimientos anteriores a la investigación.

Un aspecto fundamental de la gestión de datos para el aprendizaje automático es la rápida evolución de las fuentes de datos. Los cambios continuos en los datos pueden originarse ya sea por operaciones iniciadas por los propios ingenieros o por datos nuevos entrantes. Cualquiera de los casos requiere técnicas rigurosas de control de versiones y uso compartido de datos. Cada modelo está etiquetado con una etiqueta de procedencia que explica con qué datos se ha entrenado y qué versión del modelo. Cada conjunto de datos está etiquetado con información sobre dónde se originó y qué versión del código se usó para extraerlo y cualquier característica relacionada. Esta práctica se utiliza para asignar conjuntos de datos a modelos implementados o para facilitar el intercambio y la reutilización de datos.

5.4. Descubrimiento de conocimiento en bases de datos

Las personas tienen dificultades al enfrentarse a grandes volúmenes de datos, que en ocasiones, si no son expertos, no saben cómo manejar o tratar. En la actualidad se cuenta con una gran variedad de herramientas para coleccionar enormes cantidades de datos. Muchos de ellos han sido continuamente almacenados en bases de datos, las cuales crean una inminente necesidad y grandes oportunidades para encontrar conocimiento que puede estar inmerso en ese gran conjunto de datos y que no es visible a simple vista (Ramos et al., 2019).

El proceso completo de extraer conocimiento a partir de bases de datos se conoce como KDD (*Knowledge Discovery in Databases*). Este proceso comprende diversas etapas, que van desde la obtención de los datos hasta la aplicación del conocimiento adquirido en la toma de decisiones. Entre esas etapas, se encuentra la que puede considerarse como el núcleo del proceso KDD y que consiste en la extracción del conocimiento a partir de los datos. Esta fase es crucial para la obtención de resultados apropiados, y depende del algoritmo de aprendizaje automatizado que se aplique, no obstante, se ve influida en gran medida por la calidad de los datos que llegan desde la fase previa (Van Santen et al., 2019).

El descubrimiento de conocimiento en bases de datos está formado por un conjunto de herramientas y técnicas que facilitan la extracción de la información útil, presente en los datos, por parte de los usuarios. Este ayuda en la solución de uno de los principales problemas que la era de la información ha traído, la sobrecarga de datos.

Una adecuada extracción del conocimiento posibilita utilizar la información generada en los procesos de las organizaciones, lo cual es un elemento fundamental para mejorar la calidad y productividad de sus productos y servicios permite la utilización de la información y los datos, y el potencial de las habilidades, competencias, ideas, compromisos y motivaciones de las personas; y apoya el proceso de toma de decisiones.

El KDD está relacionado fundamentalmente con la identificación de patrones interesantes y su descripción de manera concisa y significativa. A pesar de la creciente versatilidad, los sistemas de descubrimiento de conocimiento, tienen un importante componente de interacción humana que es inherente a todo proceso de representación, manipulación y procesamiento del conocimiento (Xie et al., 2021).

El análisis de grandes volúmenes de información con el objetivo de obtener nuevo conocimiento a partir de ello supone hoy en día un importante reto. Con frecuencia se cuenta con un gran número de datos y se quiere aprender y descubrir la relación entre los mismos a partir de la aplicación de técnicas de minería de datos.

El proceso de descubrir conocimiento a partir de los datos se puede definir como el proceso no trivial de extraer y presentar conocimiento implícito, previamente desconocido, potencialmente útil y humanamente comprensible, a partir de grandes conjuntos de datos, con objeto de predecir de forma automatizada tendencias y comportamientos de los datos y/o descubrir de forma automatizada modelos previamente desconocidos.

El KDD se puede abordar, en función del problema a resolver, desde dos perspectivas distintas: desde el punto de vista predictivo, en el que se intenta obtener conocimiento para clasificación o predicción, o desde el punto de vista descriptivo, como un proceso inducción descriptiva

cuyo objetivo fundamental es descubrir conocimiento de interés dentro de los datos, intentando obtener información que describa el modelo que existe detrás de los datos. Entre las técnicas más conocidas para abordar cada una de las perspectivas, anteriormente descritas, pueden mencionarse:

- **Descriptivas**: agrupamiento o clustering, reglas de asociación, análisis de patrones secuenciales, análisis de componentes principales, detección de desviación.
- **Predictivas**: regresión y clasificación (árboles de decisión, clasificación bayesiana, redes neuronales, algoritmos genéticos, conjuntos y lógica difusa).

5.5. Repositorios de datos para la extracción de conocimiento

Los repositorios de datos permiten la compartición, reutilización y localización de los datos para el aprendizaje y descubrimiento del nuevo conocimiento dentro de las organizaciones. Los datos deben estar limpios y en el formato idóneo para ser analizados. La extracción de información de estos datos se puede utilizar para crear varias aplicaciones inteligentes en los dominios relevantes. Por ejemplo, para construir un sistema de ciberseguridad inteligente y automatizado basado en datos. Por lo tanto, se necesitan con urgencia herramientas y técnicas de gestión de datos que tengan la capacidad de extraer información o conocimiento útil de los datos de manera oportuna e inteligente, en las que se basan las aplicaciones del mundo real.

Usualmente se evalúa la calidad del conocimiento resultante de aplicar algún método de aprendizaje usando el conjunto de control; es decir, la evaluación es post-aprendizaje. A partir de los datos disponibles se aplican diferentes métodos de aprendizaje para determinar cuál produce un mejor conocimiento. El estudio de la relación entre el conjunto de entrenamiento y la eficiencia y eficacia lograda en el proceso de aprendizaje se realiza usando el método de prueba y error de una forma experimental. Es decir, se realizan sucesivos procesos de entrenamiento y se validan los resultados alcanzados. De modo que resulta de gran interés poder estimar la calidad de los datos antes de proceder al aprendizaje para evitar trabajos innecesarios.

Sin embargo, tanto los métodos de aprendizaje como los métodos para mejorar estos conjuntos dependen de la información original contenida en ellos. El estudio de calidad del conjunto de entrenamiento puede servir de base para tomar decisiones sobre cómo desarrollar el aprendizaje (Piñero et al., 2020). En el área del aprendizaje automático y la ciencia de datos, los investigadores utilizan varios conjuntos de datos ampliamente utilizados para diferentes propósitos.

5.6. Minería de datos

La Minería de Datos (conocida por su nombre en inglés *Data Mining*) es una etapa dentro del proceso de KDD en la que se estudian los datos y se extraen, a través de un conjunto de técnicas y herramientas, información útil oculta en ellos. Puede afirmarse que es la aplicación de algoritmos específicos para extraer patrones interesantes en los datos. La Minería de Datos (DM, por sus siglas en inglés) es entendida como el proceso de descubrir conocimientos interesantes, como patrones, asociaciones, cambios, anomalías y estructuras significativas a partir de grandes cantidades de datos almacenadas en bases de datos, *datawarehouses*, o cualquier otro medio de almacenamiento de información (Castrillón et al., 2022). La aplicación

de algoritmos de minería de datos requiere de actividades previas destinadas a preparar los datos de manera homogénea. Esta primera etapa es también conocida como ETL (*Extract, Transform and Load*). Un proceso completo de aplicación de técnicas de minería, mejor conocido como proceso de descubrimiento del conocimiento en bases de datos establece a la minería de datos como una etapa del mismo (Dogan & Birant, 2021).

Entendemos en el contexto de esta investigación que la **minería de datos** constituye un campo inter-disciplinario donde convergen técnicas estadísticas, matemáticas, inteligencia artificial, el reconocimiento de patrones y las bases de datos. Consiste en la extracción no trivial de información que reside de manera implícita en los datos

Dentro del procedimiento de extracción de conocimiento de bases de datos (KDD), la MD es la etapa que realiza el descubrimiento de conocimiento mediante la aplicación de procedimientos automáticos o semiautomáticos. Para ello se utilizan métodos de Inteligencia Artificial, Aprendizaje Automático, estadística y *Soft Computing*.

Un patrón llega a ser interesante si se comprende fácilmente por humanos, válido en nuevos datos o pruebas con algún grado de certeza, novedoso, potencialmente útil, o valida una hipótesis que se desea confirmar.

El uso de la Minería de Datos está relacionado con los siguientes objetivos:

- **Predicción**: predecir posibles situaciones futuras sobre la base de los acontecimientos anteriores.
- **Descripción**: explicar las razones por la que ocurren algunos eventos
- **Verificación**: examinar la existencia de algún tipo de relación entre entidades.
- **Detección de excepciones**: Detectar situaciones (registros) en las bases de datos que se corresponden a algo inusual.

Cuando se realiza un proceso de minería de datos, es necesario tener en cuenta el conocimiento previo; este puede derivar del proceso mismo: elección de variables, técnicas, algoritmos, interpretación de resultados, o del dominio de aplicación.

En la actualidad existen un conjunto de técnicas de Minería de Datos que permiten afrontar diferentes tipos de problemas, entre los que se encuentran: clasificación, *clustering*, regresión o búsqueda de reglas de asociación. A continuación se describen los diferentes tipos de problemas que permiten solucionar las técnicas de Minería de Datos.

5.6.1. Problemas de Clasificación

Las técnicas de Clasificación permiten clasificar a un individuo nuevo dentro de un conjunto predefinido de clases. Existen diferentes enfoques dentro de las técnicas de clasificación. Todos ellos, sin embargo, construyen un modelo de clasificación a partir de un conjunto de individuos de entrenamiento de los que se conocen ciertos atributos, incluyendo la clase a la que pertenecen. Ante un nuevo individuo sin clasificar, el modelo de clasificación determinará su clase haciendo uso del valor conocido de sus atributos (Martínez-Plumed et al., 2019).

- **Árboles de decisión:** Se basan en la construcción iterativa de un árbol de decisión (también llamado árbol de clasificación). Para clasificar individuos, los nodos de los árboles de decisión almacenan una condición sobre un determinado atributo del

individuo a clasificar, mientras que las ramas son las que determinan a qué nodo del nivel inmediatamente inferior descender dependiendo de que se cumpla o no dicha condición. Este proceso se realiza hasta alcanzar un nodo hoja, que almacena la clase en la que es clasificado el individuo.

- **Clasificación Bayesiana:** La clasificación Bayesiana es una técnica de aprendizaje probabilística que calcula hipótesis probabilísticas explícitas. En este caso, las hipótesis que se trabajan son las de pertenencia del individuo sin clasificar a cada una de las clases. Aquella hipótesis con mayor probabilidad será la que determine la clase del individuo que se desea clasificar.
- **Redes de Neuronas Artificiales:** Las redes de neuronas artificiales son una técnica que modela computacionalmente el aprendizaje humano llevado a cabo a través de las neuronas del cerebro. Una red neuronal se compone de unidades llamadas neuronas conectadas entre sí. Las redes de neuronas poseen una serie de características que las hacen muy interesantes entre las cuales destacan:

5.6.2. Problemas de Regresión

Las técnicas de Regresión tienen como objetivo predecir el valor (desconocido) de un atributo de un determinado individuo a partir de los valores (conocidos) de otros atributos de dicho individuo. Existen dos tipos de técnicas de regresión dependiendo del tipo de modelo de regresión generado: lineal y no lineal (Leal-Cornejo et al., 2019).

En ocasiones puede que las variables no presenten una dependencia lineal, por lo que han de emplearse técnicas de regresión no lineal, en las cuales la relación entre variables puede ser polinómica.

5.6.3. Problemas de Clustering

Las técnicas de *Clustering* pretenden dividir los objetos en grupos (denominados clusters), atendiendo a sus características y/o comportamientos. El objetivo principal de las técnicas de clustering es realizar una partición de los datos de forma que los elementos que pertenecen a un mismo cluster sean muy similares entre sí y los elementos de clusters diferentes sean lo más diferentes posible. Para lograrlo es importante elegir la medida de distancia adecuada. A diferencia de las técnicas de clasificación, el clustering es un proceso de aprendizaje no supervisado ya que las clases no están predefinidas sino que deben ser descubiertas dentro de los datos (Ruiz Rivera & Ruiz Lizama, 2021).

5.6.4. Problemas de Reglas de Asociación:

La minería de Reglas de Asociación es el proceso de búsqueda de patrones frecuentes, asociaciones, correlaciones, o estructuras causales entre los conjuntos de elementos u objetos de bases de datos de transacciones, bases de datos relacionales y otros repositorios de información. En un conjunto de transacciones, donde cada transacción es un conjunto de literales llamados ítems, una regla de asociación es una implicación de la forma X => Y; es decir, si todos los ítems de X existen en una transacción entonces todos los ítems de Y, con una alta probabilidad, están en la transacción, y donde X e Y no tienen ningún ítem en común (Vera et al., 2019).

5.6.5. Herramientas de minería de datos más utilizadas

Knime: *Konstanz Information Miner* (Knine) es un marco de trabajo gráfico para desarrollar procesos de análisis como: transformación de datos, análisis predictivos, visualización de los datos y reportes. La plataforma inicialmente desarrollada por la Universidad de Konstanz, Alemania, provee cerca de 1000 módulos para el trabajo de minería de datos. Permite realizar muestreos, transformaciones, agrupaciones de los datos. Así como su visualización a través de histogramas. Incluye la validación de modelos a partir de curvas ROC y algoritmos minería de datos tales como árboles de decisión, máquinas de soporte vectorial, regresiones, entre otros. Incluye funcionalidades adicionales como el uso de repositorios compartidos, autenticación, ejecución remota, programación, y una interfaz de usuario en la web (Torres & Cardenas, 2021).

RapidMiner: RapidMiner, anteriormente conocido como YALE (por sus siglas en inglés: Yet *Another Learning Environment*) es un programa informático multiplataforma, desarrollado sobre el lenguaje Java, para el análisis y minería de datos. Actualmente se publica bajo los términos de la Licencia AGPL (László & Ghous, 2020). Permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores a través de un entorno gráfico. Proporciona más de 500 operadores orientados al análisis de datos, incluyendo los necesarios para realizar operaciones de entrada y salida, pre-procesamiento de datos y visualización. Permite la integración con Weka, a través del uso de los algoritmos incluidos en este último y con otros programas a través de llamadas a las bibliotecas de RapidMiner. Puede ser usado mediante la interfaz de usuario que posee o en líneas de comandos. Muestra una representación interna de los procesos de análisis de datos en ficheros XML.

Weka: Desarrollado por la Universidad de Waikato, es una colección de algoritmos para el análisis de datos, que pueden ser aplicados directamente al conjunto de datos o llamados desde otras aplicaciones implementadas en Java. Contiene una colección de herramientas de visualización, pre-procesamiento, clasificación, regresión, clustering, reglas de asociación, que pueden ser accedidos a través de una interfaz gráfica. Permite el acceso a datos almacenados en bases de datos mediante conexión SQL. Actualmente se distribuye bajo la licencia GNU (*General Public License*) (Merlini & Rossini, 2021).

R: R es un lenguaje y entorno de programación, multiplataforma distribuido bajo la Licencia GPL, para análisis estadístico y gráfico que brinda la posibilidad de cargar diferentes bibliotecas o paquetes con finalidades específicas de cálculo, análisis de datos o gráfico. R proporciona un amplio abanico de herramientas estadísticas (modelos lineales y no lineales, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, etc.) y gráficas. R forma parte de un proyecto colaborativo y abierto. Existe un repositorio oficial de paquetes cuyo número supera los 4000, éstos se han organizado en vistas (o temas), que permiten agruparlos según su naturaleza y función (Rojas, 2020).

SPSS Modeler: IBM SPSS Modeler es un marco de trabajo para la minería de datos que permite elaborar modelos predictivos, descubrir patrones y tendencias en datos estructurados o no estructurados. Posee una interfaz visual soportada por análisis avanzado. Brinda la posibilidad de modelar los resultados y conocer los factores que influyen en ellos (Wendler &

Gröttrup, 2021). Incluye un área de trabajo de análisis de textos completamente integrada para analizar el texto de documentos, mensajes de correo electrónico, blogs, hilos RSS y otras fuentes de texto. Funciona con una amplia variedad de plataformas, bases de datos, hojas de cálculo y archivos sin formato.

5.6.6. Minería de datos anómalos

La minería de datos anómalos (*outliers mining*) es una disciplina dentro de la minería de datos. Se define dato anómalo como una observación que se desvía mucho del resto de las observaciones apareciendo como una observación sospechosa que pudo ser generada por mecanismos diferentes al resto de los datos. El dato anómalo, es una observación que se desvía marcadamente de otros miembros de la muestra en la cual se encuentra. Esta área de la minería de datos ha tenido aplicación en disímiles escenarios entre los que destacan: la detección de fraudes en tarjetas de créditos, análisis de irregularidades en procesos de votación, en la detección de intrusos en redes, en la detección de anomalías en imágenes médicas, en la detección de fraude en las telecomunicaciones y la industria (Pérez Esteras, 2020).

5.7. Big Data

El análisis de Big Data (BD) es una clave esencial para la fabricación digital, ya que actúa como habilitador de tecnologías. Además, el alcance de la personalización masiva que se centra en las necesidades de los mercados individualizados utiliza el análisis de BD como base. BD es un término que describe grandes volúmenes de datos variables, complejos y de alta velocidad que requieren técnicas avanzadas y técnicas para permitir la captura, el almacenamiento, la distribución, la gestión y el análisis de la información (Ngiam & Khor, 2019).

BD exige formas rentables e innovadoras de procesamiento de información para obtener mejores conocimientos. De acuerdo con las definiciones investigadas de BD, a diferencia del procesamiento de datos tradicional, la primera sugerencia para caracterizar BD se relacionó en términos de Volumen, Variedad y Velocidad, también denominadas como las **Tres V**. Estas fueron las tres dimensiones que surgieron como un marco común de desafíos en la gestión de datos.

Procesar continuamente grandes cantidades de datos heterogéneos no estructurados recopilados en formatos como video, audio, texto u otros, adicionalmente, también se han intentado asignar otras dimensiones para una mejor caracterización como: Veracidad, Visión, Volatilidad, Verificación, Validación, Variabilidad y Valor. Según varios autores la descripción de las dimensiones es la siguiente:

- **Volumen**: gran tamaño de volumen de datos que consume gran capacidad de almacenamiento o consta de una enorme cantidad de colecciones. Los tamaños de BD se mencionan en múltiples terabytes y petabytes;
- **Variedad**: varios tipos de datos, generados a partir de una gran variedad de fuentes y formatos, y contenidos de campos de datos multidimensionales. Se refiere a la heterogeneidad estructural en un conjunto de datos;

- **Velocidad**: producción rápida. Generación, análisis, entrega y creación de datos medidos por su frecuencia. Se refiere a la tasa de generación de datos y la velocidad para analizar y actuar;
- **Veracidad**: representa la falta de confiabilidad en algunas fuentes de datos. Algunos datos requieren un análisis de BD para obtener una predicción confiable;
- **Visión**: solo un proceso con un propósito debe enviar la generación de datos. La probabilidad del proceso de generación de datos se aborda en esta dimensión;
- **Volatilidad**: una vida útil limitada puede caracterizar los datos generados. Esta dimensión aborda el concepto del ciclo de vida de los datos. Asegura la reposición de los datos obsoletos con datos nuevos;
- **Verificación**: conformidad de los datos generados por un conjunto de especificaciones. Asegura la conformidad de las medidas de ingeniería;
- **Validación** – la conformidad de la visión de los datos generados. Detrás del proceso, se asegura la transparencia de supuestos y conexiones;
- **Variabilidad** – caudales de datos medidos por su variación. Se agregaron Variabilidad y Complejidad como dos dimensiones adicionales de BD;
- **Valor**: a través de la extracción y la transformación, define hasta qué punto BD genera conocimientos y beneficios económicamente valiosos. Valor como atributo definitorio de BD.

5.8. Conclusiones del capítulo

Teniendo en cuenta la gran afluencia de datos, definitivamente es necesario mejorar la forma en que se diseñan y desarrollan los modelos de datos computacionales/analíticos convencionales. El modelado de datos sostenible se puede definir como una forma de tecnología de modelado de datos, cuyo objetivo es dar sentido a la gran cantidad de datos asociados en su propio campo, mediante el descubrimiento de patrones y correlaciones de una manera eficaz y eficiente. El modelado de datos sustentable se enfoca específicamente en la máxima precisión de aprendizaje con un costo computacional mínimo; y el procesamiento rápido y eficiente de grandes volúmenes de datos.

El modelado de datos sostenible parece ser ideal debido a la facilidad con la que grandes cantidades de datos se manejan de manera eficiente, así como a la reducción de costos asociada que se observa en muchos casos. Desde una perspectiva más amplia, implica una revolución en el modelado de datos en las ciencias electrónicas.

BD ofrece la oportunidad de mutar del paradigma de fabricación actual a la fabricación inteligente. Por lo tanto, BD puede ayudar a los fabricantes a tomar decisiones más racionales, informadas y receptivas. Estas características de BD mejoran la competitividad de la fabricación en el mercado mundial. Los datos de IoT convergen en BD para analizarlos y sacar conclusiones de los conjuntos de datos recopilados. En otras palabras, los datos de IoT serán parte de BD y BD no se puede explorar más sin IoT.

CAPÍTULO VI: SEGURIDAD DE SISTEMAS COMPUTACIONALES

6.1. Introducción del capítulo

Cada año son más los dispositivos que se conectan a la red global Internet. Los ataques de red como el *ransomware*, el robo de identidad, la denegación de servicio, el robo de datos y los ataques de día cero, son difíciles de rastrear utilizando los mecanismos de seguridad estándar, como el *firewall* y el software antivirus. Asociado a esto, el IoT, los entornos virtuales, el acceso remoto, los datos almacenados en sistemas en la nube, constituyen oportunidades abiertas que representan nuevas vulnerabilidades cada vez mayores que conducen a una información comprometida para las personas y las empresas.

La Ciberseguridad (CS) Ha sido definida como un nuevo término en un alto nivel de seguridad de la información. CS es una tecnología que se basa en proteger, detectar y responder a los ataques (Ospina Díaz & Sanabria Rangel, 2020). El aumento de dispositivos conectados significa más posibilidades de ciberataques. Las razones principales de ataques son las siguientes:

- Dispositivos funcionando durante demasiado tiempo sin actualizar las herramientas de seguridad o antivirus.
- Número considerable de controladores antiguos utilizados en redes de Sistemas de Control Industrial (ICS), diseñados cuando la CS no era una preocupación.
- Las amenazas de CS pueden ingresar pasando por alto las medidas de CS debido a la existencia de múltiples vías desde varias redes de ICS.
- Rápida propagación de *malware* debido a varias redes ICS que aún permanecen implementadas como una red plana sin aislamiento físico o virtual entre otras redes no relacionadas.

Las amenazas cibernéticas afectan:

1) **Confidencialidad**: necesaria para mantener la seguridad de los datos personales del usuario en el CPS y evitar que un atacante intente cambiar el estado del sistema físico "escuchando" los canales de comunicación entre los sensores y el controlador, y entre el controlador y el actuador.
2) **Integridad**: cuando los datos o recursos se pueden cambiar sin permiso.
3) **Disponibilidad**: cuando existan fallas en la tecnología informática, de gestión, comunicación, equipamiento.
4) **Confiabilidad**: cuando es necesario confirmar que ambas partes involucradas son realmente quienes pretenden ser.
5) **Autenticidad:** cuando se pueda probar que la identidad de un sujeto o recurso es el único reclamo.
6) **No repudio**: cuando se puede probar que los hechos o ataques han tenido lugar para que no puedan ser repudiados posteriormente.
7) **Rendición de cuentas**: cuando las acciones de una entidad pueden atribuirse únicamente a la entidad.

6.2. Vulnerabilidades de los sistemas computacionales

Hay muchos eventos de ataque a la red ocurridos en estos años mediante la explotación de vulnerabilidades. Estas vulnerabilidades pueden estar en el sistema operativo, el software de la

aplicación, la debilidad en el hardware del servidor de la computadora, configuraciones inadecuadas de autenticación en el sistema o mal uso de los usuarios. Los efectos de las vulnerabilidades hacen que los sistemas sean accedidos, modificados, apagados fuera de la política de seguridad en las organizaciones (Asatiani et al., 2021).

Las vulnerabilidades de la computadora son el resultado de fallas en el diseño del software, hardware y firmware, o incluso de una configuración o configuración incorrecta en estos sistemas. Las vulnerabilidades conducen a resultados imprevistos que pueden reducir el rendimiento de los sistemas o dañar los datos de los sistemas. En algunos casos, los datos sensibles pueden revelarse o duplicarse aprovechando las vulnerabilidades de estos sistemas. Las vulnerabilidades afectan a los sistemas de forma diferente con características diferentes. Para identificar vulnerabilidades, cada vulnerabilidad encontrada se etiqueta con un número de referencia (Lai & Hsia, 2007).

6.3. Categorías de ataque

Los analistas de seguridad de redes pueden detectar intrusiones observando la información obtenida de los paquetes de red a través del análisis de flujo de red. Dentro de las categorías de ataques, se pueden enumerar las siguientes (Alguliyev et al., 2018):

Ataque	Descripción
Denegación de servicio. (DoS, *Denial of service*)	Es un ataque de intrusión realizado al hacer que los recursos de la red estén ocupados y no disponibles para los usuarios legítimos.
Denegación de Servicio Distribuido. (DDoS, *Distributed Denial of Service*)	Inhabilitan la continuidad de comunicación corporativas con el exterior vía web, en la que se afecta no solo el ancho de banda, sino también la latencia y las tablas conmutadas de flujo de datos. Esto se logra debido a que se realizan múltiples peticiones a uno o varios servidores (web, correo electrónico, base de datos, proxy, etc.) desde diferentes lugares del mundo, con el objetivo de saturar el sistema hasta hacerlo colapsar, o efectuar ataques de fuerza bruta mediante malware especializados que escanean Internet en busca de dispositivos que estén conectados al IoT para obtener sus contraseñas y secuestrarlos.
Inyección de datos falsos (FDI, *False Data Injection*)	Puede eludir todas las técnicas convencionales de vigilancia y seguridad. Cuando tiene éxito, Los ataques FDI permiten al atacante comprometer las mediciones de los sensores de la red, obstaculizar el funcionamiento normal de una red eléctrica y, en ocasiones, incluso dañar los dispositivos conectados a ella.
User to Root (U2R)	Es un ataque de intrusión causado por obstaculizar la autenticidad del usuario al permitir el acceso root al intruso.
Remote to Local (R2L)	Es un ataque de intrusión causado por romper la integridad de la red y permitir el acceso de la red local al intruso.
Software malicioso *(Malware)*	Cualquier software que intencionalmente ejecute cargas maliciosas en las máquinas de las víctimas (computadoras, teléfonos inteligentes, redes informáticas, etc.) se considera malware. Existen diferentes tipos de malware, incluidos virus, gusanos, troyanos, rootkits y ransomware. Cada tipo y familia de malware está diseñado para afectar la máquina de la víctima original de diferentes maneras, como dañar el sistema de destino, permitir la ejecución remota de código, robar datos confidenciales (Aslan & Samet, 2020).

Man-in-The-Middle (MiTM)	Es un término general para cuando un perpetrador se posiciona en una conversación entre un usuario y una aplicación, ya sea para escuchar a escondidas o para hacerse pasar por una de las partes, haciendo que parezca un intercambio normal de información está en marcha. El objetivo de un ataque es robar información personal, como credenciales de inicio de sesión, detalles de cuentas y números de tarjetas de crédito. Los objetivos suelen ser los usuarios de aplicaciones financieras, empresas, sitios de comercio electrónico y otros sitios web en los que es necesario iniciar sesión.
Sybil	Un ataque de Sybil utiliza un solo nodo para operar muchas identidades falsas activas (o identidades de Sybil) simultáneamente, dentro de una red de igual a igual. Este tipo de ataque tiene como objetivo socavar la autoridad o el poder en un sistema respetable al obtener la mayoría de la influencia en la red. Las identidades falsas sirven para proporcionar esta influencia. Un ataque exitoso de Sybil proporciona a los actores de amenazas la capacidad de realizar acciones no autorizadas en el sistema.
Black hole	Los ataques de agujero negro ocurren cuando un enrutador elimina todos los mensajes que se supone que debe reenviar. De vez en cuando, un enrutador está mal configurado para ofrecer una ruta de costo cero a todos los destinos en Internet. Esto hace que todo el tráfico se envíe a este enrutador. Dado que ningún dispositivo puede soportar tal carga, el enrutador falla.
Wormhole	El ataque de agujero de gusano es un ataque grave y popular. Este ataque involucra a dos o más de dos nodos maliciosos y el paquete de datos de un extremo del nodo malicioso se canaliza al otro nodo rencoroso/malicioso en el otro punto, y estos paquetes de datos se transmiten.
Sonda	Es una actividad de intrusión realizada mediante el escaneo de la red y la recopilación de toda la información relacionada con la red sobre las actividades de la red realizadas en la red.

Los ataques de inyección de datos falsos (FDI) son uno de los ataques ciberfísicos más destructivos contra las redes inteligentes. Los ataques FDI comprometen las mediciones en el Sistema de Control de Supervisión y Adquisición de Datos (SCADA), que tienen como objetivo manipular el voltaje estimado por la estimación de estado en el sistema de gestión de energía del sistema de potencia. Los ataques FDI pueden causar graves consecuencias, como sobrecarga de línea, desconexión de carga, estados inestables del sistema e incluso colapso de voltaje.

El ataque de repetición tiene como objetivo los protocolos relacionados con la autenticación y el intercambio de claves para capturar o almacenar una sesión completa o un fragmento de una sesión en el tráfico legítimo. Al ganar confianza en la red pública, el atacante envía el mensaje capturado para indicar la participación en la sesión original. Un ataque de repetición se menciona como una debilidad o vulnerabilidad de seguridad en el procedimiento de autorización para acceder a los datos almacenados (Rodriguez & Celis, 2019).

El ataque de adivinación de contraseñas ocurre al escuchar el canal de comunicación al explorar las debilidades en numerosos protocolos de autenticación. Este tipo de ataque podría tener lugar en modo en línea o fuera de línea. El atacante podrá adivinar todas las contraseñas posibles para tener éxito en el proceso de autenticación. El objetivo principal de un ataque de

suplantación de identidad es hacer que los servidores confíen en que el atacante es una de las entidades autorizadas.

Los ataques de DDoF, malware y Man-in-The-Middle se centran en violar las restricciones de confidencialidad y disponibilidad y hacer que los dispositivos IoT y los canales de comunicación sean vulnerables. Otros tipos de ataques que pueden interrumpir el funcionamiento normal de los dispositivos IoT y sus aplicaciones incluyen denegación de servicio, interferencia, Sybil, agujero negro, agujero de gusano y ataques de malware.

Una de las amenazas más peligrosas que ocurren debido a un medio de comunicación inseguro es el ataque Man-in-the-Middle (MiTM). Al instalar malware o al cambiar la funcionalidad del dispositivo, MiTM se inicia si el dispositivo no utiliza ningún mecanismo de encriptación o autenticación. Los ataques MiTM son una amenaza importante para la seguridad de los sistemas IoT debido a su dependencia de medios de comunicación inseguros. Estos ataques implican que el atacante intercepte y altere la comunicación entre dos partes sin su conocimiento o consentimiento.

Los ataques a las aplicaciones tienen como objetivo las vulnerabilidades en el software o el firmware de un dispositivo IoT. Esto puede incluir ataques de inyección SQL, secuencias de comandos entre sitios y ataques de inyección de comandos. La intrusión física implica acceder físicamente a un dispositivo IoT para extraer información, instalar malware o interrumpir su funcionamiento. Un atacante puede usar herramientas como la selección de bloqueo o el acceso físico al dispositivo IoT para realizar estos ataques.

Según estudios recientes, el *malware* está aumentando a un ritmo alarmante, y algunos malware pueden ocultarse en el sistema mediante el uso de diferentes técnicas de ofuscación. Para proteger los sistemas informáticos e Internet del malware, es necesario detectar el malware antes de que afecte a una gran cantidad de sistemas. El malware es un programa dañino que infecta los sistemas informáticos, elimina archivos de datos y roba información valiosa. El malware puede atacar los sistemas informáticos personales y de la organización. Malware es una contracción de software malicioso, está diseñado para destruir sistemas y programas informáticos. Tiene muchas formas, como virus, gusanos, troyanos y spyware. Cada año, muchos sistemas informáticos de todo el mundo se dañan como resultado de malware.

Virus: El virus es uno de los tipos de malware que es un fragmento de código que se adjunta a un programa o archivo. Cuando el programa infectado es ejecutado por un usuario, el virus se ejecuta en secreto sin que los usuarios se den cuenta (Chuquilla et al., 2019). Muchos virus necesitan cuatro etapas para infectar y destruir los sistemas informáticos:

1. La fase latente es una etapa conocida como paso inactivo porque el virus está inactivo y se activa por fecha u otro programa.
2. El virus intenta copiarse a sí mismo en otro programa en la fase de propagación.
3. En la fase de activación, en este paso el virus está listo para realizar sus funciones.

Los virus dañan los programas, borran archivos y luego apagan o reinician los sistemas informáticos en la fase de ejecución, la cuarta etapa. Estos pasos se cambian de una computadora a otra y de un sistema operativo a otro. También depende de los tipos de puntos

vulnerables en el sistema. Hay muchos tipos de virus como virus de macro, de arranque, de infección de archivos y psicológicos.

Wormhole: Se define gusano como un programa que se ejecuta independientemente de otros programas, se replica a sí mismo y se propaga a través de una red de computadora a computadora, lo que significa que el gusano es un programa dañino que infecta de host a host a través de un agujero vulnerable y de seguridad en los sistemas. La principal diferencia entre virus y gusanos es que los virus siempre se esconden en los programas, sin embargo, los gusanos funcionan de forma independiente. Además, los piratas informáticos utilizan principalmente los gusanos en lugar de los virus porque los gusanos se propagan de una computadora a otra a través de las conexiones de red. El gusano usa algunas formas para propagarse. En primer lugar, utiliza funciones de correo electrónico para copiarse de un sistema a otro. En segundo lugar, los métodos de ejecución ayudan al gusano a ejecutarse en otros sistemas (Jafferis et al., 2022). Después de eso, consume las funciones de inicio de sesión para duplicarse de un sistema a otro diferente.

Programas espía y publicitarios: El software espía se coloca en una computadora, recopila la actividad en línea y fuera de línea de los usuarios y la devuelve a la fuente central, mientras que el software publicitario se instala en una computadora con el permiso del usuario y muestra anuncios o ventanas emergentes con fines de marketing (Anumula & Raymond, 2022).

Ransomware: Ransomware es un software malicioso que utilizan los ciberdelincuentes para cifrar y bloquear computadoras, teléfonos inteligentes y dispositivos de datos; pide a los usuarios del dispositivo que paguen el rescate para desbloquear o descifrar sus dispositivos. Se propaga a través de enlaces de correo electrónico y archivos adjuntos, así como a través de sitios web infectados y discos USB (Niño, 2023). Este malware se instala en dispositivos móviles en forma de aplicaciones y se descarga desde un sitio web de terceros, no desde las tiendas oficiales.

Trojan: Trojan es un programa que los piratas informáticos instalan en un sistema. Copia información sin la autorización del usuario. Los usuarios conocen los procesos de instalación de los troyanos, pero desconocen sus procesos ocultos (Musgrave et al., 2020). Los atacantes usan troyanos para propagar virus u otros tipos de malware en los sistemas sin la atención del usuario.

El éxito del ataque cibernético se basa en la información que posee un atacante cuando se lanza el ataque y, a menudo, se mide por la información obtenida o modificada como resultado del ataque. Por lo tanto, la información debe ser un elemento esencial de cualquier teoría de los ciberataques.

6.4. Protocolos de defensa

Hoy en día, se puede ver que tanto los usuarios como las actividades en línea han crecido significativamente. Considerando que, no todas las activaciones de los usuarios son apropiadas, como resultado, se deben utilizar mecanismos de protección para proteger los sistemas electrónicos del acceso no autorizado.

Tabla 6. Sistemas y protocolos de defensa.

Defensa	Descripción
SNMP: Protocolo simple de administración de red.	El SNMP se usa para monitorear de cerca los patrones anormales del tráfico de red entrante en función de los cambios en los valores de las variables de la Base de Información de Administración (MIB) de SNMP (Ravindran et al., 2023).
Kullback–Leibler distance (KLD)	La detección de inyección de datos falsos se realiza mediante la estimación de la distancia Kullback-Leibler (KLD) entre los datos originales (en el momento del envío) y los datos recibidos (después de enviarlos a través del canal de comunicación). Se ha demostrado que este método es más eficaz en la detección de tráfico anómalo en función del tamaño del paquete o de las estadísticas de distribución. Se realizan distribuciones de probabilidad para los datos antes del envío en la red y después del envío en el canal de comunicación, y se analizan las diferencias entre ellos.
ACL: Reglas de control de acceso	Las ACL se utilizan para filtrar y asegurar el tráfico de las redes. Las ACL filtran el tráfico de la red controlando si los paquetes enrutados o conmutados se han enviado o bloqueado en la interfaz. Estos paquetes son examinados para determinar cómo deben manipularse en función de los criterios establecidos por la ACL
MTD: Defensa de objetivos móviles.	MTD se han promocionado como un enfoque innovador para la seguridad informática que elimina la naturaleza estática de los sistemas informáticos actuales (la mayor ventaja de un atacante). Si bien es prometedor, el dinamismo de MTD presenta desafíos relacionados con la comprensión y cuantificación del impacto de los sistemas MTD en la seguridad, los usuarios y los atacantes. Para analizar este impacto, se deben formalizar tanto los conceptos de sistemas MTD como los de ciberataques (Giraldo et al., 2023).
HMTD: Defensa de objetivos móviles ocultos	HMTD) es una estrategia de defensa proactiva que se mantiene oculta de los atacantes al cambiar la reactancia de las líneas de transmisión para frustrar los ataques de inyección de datos falsos (FDI) (Liu et al., 2023).
Cortafuegos	Un *firewall* es uno de los mecanismos de protección que se puede utilizar para controlar y monitorear los tráficos de red entrantes y salientes en función de las reglas de seguridad y tiene dos tipos de hardware y software. El cortafuegos analiza los correos electrónicos entrantes y ayuda a los servicios del sistema operativo a distinguir aplicaciones falsas y usuarios falsos. De acuerdo con una computadora personal puede protegerse con un firewall de software, pero una organización grande puede protegerse con un firewall de hardware para permitir/denegar acciones internas y externas que provienen de Internet.
Software de seguridad	Hay muchos tipos de software de seguridad, como herramientas de eliminación, antivirus y software de seguridad de Internet, que se pueden usar para proteger los sistemas informáticos contra el malware.
Antivirus	Las funciones principales de los programas antivirus, incluida la exploración de archivos de inicio, la exploración de actividades en tiempo real, como la descarga de archivos, la supervisión de actividades de aplicaciones, como navegadores web, la exploración de discos duros en busca de malware conocido, la identificación y la eliminación de malware.

Las técnicas SNMP y KLD se utilizan para detectar DDoS y ataques de intercambio de datos falsos, mientras que las técnicas ACL y MTD se aplican para mitigar estos ataques fortaleciendo el objetivo y reduciendo la superficie de ataque. La detección de DDoS se lleva a cabo analizando los patrones de ataque entrantes y la estructura de paquetes de las solicitudes de ataque y formulando reglas de ACL para bloquear patrones de DDoS conocidos (Gayathri et al., 2023).

6.5. Sistemas de Detección de Intrusos

La investigación en el campo de la seguridad cibernética ha planteado la necesidad de abordar el tema de los delitos cibernéticos que han provocado la requisición de la propiedad intelectual, como fallas en los sistemas informáticos, deterioro de datos importantes, comprometiendo la confidencialidad, autenticidad e integridad de la información de los usuarios. Teniendo en cuenta estos escenarios, es fundamental asegurar los sistemas informáticos y al usuario mediante un Sistema de Detección de Intrusos (IDS, por sus siglas en inglés) (Thakkar & Lohiya, 2020). Los IDS se implementan para examinar la información que fluye a través de la red y generar una alarma por las probables actividades maliciosas generadas por los intrusos (Mendonça et al., 2023). Los IDS pueden ser de dos tipos:

- IDS basado en firmas: detecta las intrusiones extrayendo las firmas de los paquetes de red o analizando los patrones de ataque. Genera una alerta para los patrones de firma coincidentes almacenados en la base de datos de firmas.
- IDS basado en anomalías: detecta ataques en función de los patrones de ataque

Independientemente del tipo, la arquitectura básica de IDS consta de cuatro pasos:

1. Los paquetes de red se capturan mediante sensores de red o herramientas de rastreo de red.
2. Los datos capturados luego se filtran y examinan. El filtrado se realiza en función de las reglas de filtrado.
3. Los patrones de firma se comparan con la base de datos de firmas ya disponible.
4. El IDS genera una alerta cuando se encuentra una coincidencia con la base de datos de firmas almacenada.

La detección de intrusiones es un problema de clasificación, en el que se aplican varias técnicas de *Machine Learning* (ML) y *Data Mining* (DM) para clasificar los datos de la red en tráfico normal y de ataque. Las técnicas de ML y DM implementadas en los conjuntos de datos de IDS contienen datos etiquetados y características de tráfico de red. Estos ayudan al clasificador a aprender diferentes patrones de ataque para detectar un ataque en particular.

Las características del conjunto de datos ayudan al clasificador a aprender los patrones de tráfico normales, así como los patrones de ataque a través de los cuales el clasificador puede clasificar los datos de entrada. El conjunto de datos que se usa para entrenar al clasificador se construye al monitorear el tráfico de la red durante un intervalo de tiempo particular. El conjunto de datos consta de tráfico de red normal y tráfico de red anómalo que ayuda al clasificador a identificar los patrones de los datos con una cantidad suficiente de ejemplos. Los datos recopilados se dividen en un conjunto de entrenamiento y un conjunto de prueba para entrenar y probar el clasificador, respectivamente.

Hay varias de técnicas variadas que se han utilizado con éxito para IDS. Sin embargo, cada uno de los algoritmos utilizados para IDS entrena y prueba el conjunto de datos de una manera diferente. Los algoritmos utilizados tienen instancias de red agrupadas como conjunto de entrenamiento y conjunto de prueba. La eficiencia del sistema desarrollado se puede probar y comparar en función de factores como la optimización de parámetros, la optimización de características y la variabilidad en el tamaño del conjunto de datos.

Se puede desarrollar un conjunto de datos de detección de intrusos mediante la recopilación de información de diversas fuentes como: los flujos de tráfico de red que contienen información sobre el host; el comportamiento del usuario; y las configuraciones del sistema. Esta información es necesaria para estudiar los patrones de ataque y la actividad anormal de varios ataques a la red. La actividad de la red se recopila a través de un enrutador o conmutador de red. Después de recopilar el tráfico de red entrante y saliente, se realiza un análisis de flujo de red para estudiar el tráfico de red. El análisis de flujo se puede describir como el proceso de analizar la información del paquete de red, como la dirección IP de origen; la dirección IP de destino; el número de puerto de origen; el número de puerto de destino; el tipo de servicios de red, y otros.

Técnicas para el sistema de detección de intrusos

Tabla 7. Métodos de aprendizaje automático usados para los IDS.

Métodos de clasificación	Métodos de agrupamiento	Otros métodos
Decision Tree	k-means Clustering	Hybrid Classifiers
Random Forest	Fuzzy Clustering	Ensemble Classifiers: • Homogenous • Heterogenous
Naive Bayes		
Support Vector Machine		
Artificial Neural Networks		
Multi layer Percepton		

6.6. Blockchain

En la visión del IoT, los dispositivos convencionales se vuelven inteligentes y autónomos; sin embargo, existen desafíos que abordar, particularmente en el dominio de la seguridad centrado en la confiabilidad de los datos; siendo necesario generar confianza en esta enorme fuente de información entrante. Las aplicaciones de IoT tienen características muy específicas, generan grandes volúmenes de datos y requieren conectividad y energía durante largos períodos. Esto, sumado a las limitaciones en memoria, capacidad de cómputo, redes y fuente de alimentación limitada plantean una gran cantidad de desafíos.

El problema de la confianza en los sistemas de información es sumamente complejo cuando no se prevén mecanismos de verificación ni auditoría, especialmente cuando se trata de información sensible, como transacciones económicas con monedas virtuales (Guzmán et al., 2022).

Blockchain es el mecanismo que permite que las transacciones sean verificadas por un grupo de actores poco confiables. Proporciona un libro mayor distribuido, inmutable, transparente, seguro y auditable. La cadena de bloques se puede consultar de forma abierta y completa, lo que permite el acceso a todas las transacciones que se han producido desde la primera transacción del sistema, y puede ser verificada y cotejada por cualquier entidad en cualquier momento.

El protocolo ***Blockchain*** estructura la información en una cadena de bloques, donde cada bloque almacena un conjunto de transacciones realizadas en un momento dado. Los bloques están unidos entre sí por una referencia al bloque anterior, formando una cadena.

Para admitir y operar con *Blockchain*, los pares de la red deben proporcionar la siguiente funcionalidad: enrutamiento, almacenamiento, servicios de billetera y minería. De acuerdo con las funciones que brindan, diferentes tipos de nodos pueden formar parte de la red.

Blockchain se ha convertido en una tecnología clave en la transformación de la forma en que se comparte información. Para generar confianza en entornos distribuidos sin necesidad de autoridades es un avance tecnológico que tiene el potencial de cambiar muchas industrias IoT ha aprovechado tecnologías disruptivas como Big data, *Cloud computing y blockchain* para superar sus limitaciones desde su concepción.

La tecnología ***Blockchain*** se identifica como la clave para resolver los problemas de escalabilidad, privacidad y confiabilidad relacionados con el paradigma IoT

Blockchain puede enriquecer el IoT al proporcionar un servicio de intercambio confiable, donde la información es confiable y se puede rastrear. Las fuentes de datos se pueden identificar en cualquier momento y los datos permanecen inmutables en el tiempo, lo que aumenta su seguridad. En los casos en los que la información de IoT deba compartirse de forma segura entre muchos participantes, esta integración representaría una revolución clave.

6.7. Conclusiones del capítulo

Dos preocupaciones comunes en muchas de las aplicaciones emergentes de Inteligencia Artificial, como los automóviles autónomos, los vehículos aéreos no tripulados, la medicina de precisión y las tecnologías de asistencia para personas discapacitadas, son que todas tratan con datos confidenciales de los usuarios y tienen un impacto directo en la salud y la seguridad de la vida humana. Por lo tanto, los datos deben mantenerse privados. Además, los automóviles, los drones y los dispositivos implantados no deben ser alterados por agentes malintencionados. Por lo tanto, estos dispositivos informáticos también deben mantenerse seguros. Además, en muchas aplicaciones de IA, los investigadores a menudo tienen que enviar códigos o usar modelos que no entienden por completo. El código y los modelos se envían con relativamente poca comprensión de cómo o por qué funcionan. Esto dificulta razonar sobre ataques y fugas de información, y también sobre cómo se puede engañar al modelo. En general, por todas estas razones, los investigadores deben crear políticas, mecanismos y algoritmos sólidos de seguridad y privacidad para las aplicaciones emergentes de los Sistemas Computacionales.

Los enfoques basados en el comportamiento, en la verificación de modelos y en la nube funcionan bien para el *malware* desconocido y complicado; y también surgen enfoques basados

en aprendizaje profundo, dispositivos móviles e IoT para detectar una parte del *malware* conocido y desconocido.

Las principales preocupaciones de seguridad en el entorno de IoT se clasifican en implementación, privacidad, infraestructura de red, amenazas de seguridad, *malware*, autenticación y desafíos relacionados con la autorización.

CAPÍTULO VII: INGENIERÍA DE SOFTWARE

7.1 Introducción

El capítulo sobre ingeniería de software incluye información sobre los siguientes temas:

Ciclo de vida del software: se explica detalladamente cada fase del ciclo de vida del software, desde la planificación hasta el mantenimiento, y qué sucede en cada fase. También se muestran los diferentes modelos de ciclo de vida, como el modelo en cascada, el modelo en espiral, el modelo iterativo e incremental, y cómo se utilizan en la práctica.

Metodologías de desarrollo de software: se discuten las diferentes metodologías que se utilizan para desarrollar software, como el modelo en cascada, el modelo en espiral, el modelo ágil, entre otros.

Diseño de software: en este aspecto se habla de los principios de diseño de software, como la modularidad, la cohesión y el acoplamiento, y se explica cómo se aplican en la práctica. También se habla de técnicas de diseño específicas, como el diseño orientado a objetos, y se proporcionan ejemplos de cómo se aplican en la vida real.

Pruebas de software: se consideran los diferentes tipos de pruebas de software que se utilizan para garantizar la calidad del software, como pruebas unitarias, pruebas de integración, pruebas de sistema, pruebas de aceptación, entre otros.

Mantenimiento de software: se habla de la importancia del mantenimiento de software y cómo se lleva a cabo y se explican las diferentes actividades que se realizan durante el proceso de mantenimiento, como la corrección de errores, la adaptación a cambios en los requisitos y la mejora del rendimiento, y proporcionar ejemplos de cómo se han llevado a cabo estas actividades en proyectos reales

Herramientas de ingeniería de software: se discuten las diferentes herramientas de software que se utilizan en el desarrollo de software, como editores de código, compiladores, depuradores, gestores de versiones, entre otros. Y se explica cómo se utilizan estas herramientas en la práctica, y proporcionar ejemplos de cómo se han utilizado en proyectos reales

Pruebas de software: se describe los diferentes tipos de pruebas de software, como las pruebas unitarias, las pruebas de integración, las pruebas de sistema y las pruebas de aceptación. Se explica en qué consiste cada tipo de prueba, cómo se llevan a cabo y cuáles son las mejores prácticas en cada caso.

7.2 Ingeniería de software

La ingeniería de software se puede definir como la disciplina que se encarga del diseño, desarrollo, implementación, prueba y mantenimiento de software de alta calidad. Esta disciplina se basa en la aplicación de principios de ingeniería y métodos científicos para crear software eficiente, seguro, confiable y escalable.

La ingeniería de software se enfoca en abordar los desafíos de desarrollo de software a través de la aplicación sistemática y rigurosa de procesos de software. Los ingenieros de software utilizan herramientas y técnicas para gestionar los requisitos de los usuarios, diseñar arquitecturas de software, implementar código, realizar pruebas y mantener el software durante todo su ciclo de vida.

El objetivo principal de la ingeniería de software es mejorar la calidad del software y hacer que el proceso de desarrollo sea más eficiente y efectivo. Esto se logra mediante la aplicación de metodologías, prácticas y herramientas que permiten a los ingenieros de software trabajar de manera más organizada y sistemática para producir software de alta calidad.

En resumen, la ingeniería de software es una disciplina que se enfoca en aplicar la ingeniería y los métodos científicos para diseñar, desarrollar y mantener software de alta calidad. Se basa en prácticas y técnicas sistemáticas para abordar los desafíos del desarrollo de software y mejorar la eficiencia y la calidad del proceso de desarrollo.

El software, también conocido como programa de computadora o aplicación informática, es un conjunto de instrucciones que le indican a la computadora cómo realizar una tarea específica. El software se compone de programas, scripts, archivos de configuración y otros componentes que permiten que la computadora realice una amplia variedad de tareas y funciones.

El software puede clasificarse en dos categorías principales: software de sistema y software de aplicación. El software de sistema es el conjunto de programas y herramientas que permiten que la computadora funcione y controle el hardware. Algunos ejemplos de software de sistema son el sistema operativo, el controlador de dispositivos y el software de utilidades del sistema.

Por otro lado, el software de aplicación es el conjunto de programas que los usuarios utilizan para realizar tareas específicas, como procesamiento de texto, edición de imágenes, navegación por la web y juegos. Los programas de aplicación se crean utilizando lenguajes de programación como C++, Java, Python, entre otros.

En resumen, el software es un conjunto de instrucciones que permiten a la computadora realizar tareas específicas. Se compone de programas, scripts, archivos de configuración y otros componentes. Existen dos categorías principales de software: software de sistema y software de aplicación.

El software se puede clasificar en dos categorías principales: software de sistema y software de aplicación.

Software de sistema: es el conjunto de programas y herramientas que permiten que la computadora funcione y controle el hardware. Su función principal es administrar los recursos del sistema, proporcionar una interfaz de usuario y controlar el hardware. Algunos ejemplos de software de sistema son:

Sistema operativo: es el programa que controla el hardware y los recursos de la computadora. Ejemplos de sistemas operativos son Windows, macOS, Linux, etc.

Controlador de dispositivos: es el software que permite que los dispositivos externos, como impresoras, cámaras y escáneres, se comuniquen con la computadora.

Software de utilidades del sistema: son programas que realizan tareas de mantenimiento y optimización en el sistema, como la limpieza del disco duro, la desfragmentación, la gestión de archivos, etc.

Software de aplicación: es el conjunto de programas que los usuarios utilizan para realizar tareas específicas. Su función es proporcionar al usuario herramientas para realizar sus actividades y trabajos. Algunos ejemplos de software de aplicación son:

Procesadores de texto: son programas que permiten la creación y edición de documentos de texto, como Microsoft Word, Google Docs, etc.

Hojas de cálculo: son programas que permiten la creación y edición de hojas de cálculo, como Microsoft Excel, Google Sheets, etc.

Navegadores web: son programas que permiten la navegación por Internet, como Google Chrome, Mozilla Firefox, etc.

En resumen, el software se clasifica en dos categorías principales: software de sistema y software de aplicación. El software de sistema administra los recursos del sistema y controla el hardware, mientras que el software de aplicación proporciona herramientas para realizar tareas específicas.

El ciclo de vida del software es el proceso de desarrollo completo de un software, desde su concepción hasta su retirada. Este proceso se divide en varias fases, cada una con un propósito y objetivos específicos. A continuación, describiremos cada una de las fases del ciclo de vida del software y su importancia:

Análisis de requisitos: En esta fase, se identifican y analizan los requisitos del software. Es esencial que se comprendan los requisitos del software para diseñar una solución que cumpla con las necesidades del usuario. Esta fase también incluye la definición de los objetivos y el alcance del proyecto. La importancia de esta fase es crucial ya que una comprensión incorrecta o incompleta de los requisitos del usuario puede llevar a un software defectuoso y a una insatisfacción del usuario final.

Diseño: En esta fase, se define la arquitectura y el diseño del software. Se desarrolla una solución que cumpla con los requisitos del usuario y se planifica la estructura de la aplicación. El diseño también incluye la definición de interfaces de usuario y la selección de tecnologías y herramientas. La importancia de esta fase es fundamental ya que el diseño incorrecto o la falta de una planificación adecuada puede llevar a un software ineficiente y difícil de mantener.

Implementación: En esta fase, se lleva a cabo la codificación del software. Se implementa el diseño del software y se crean los componentes del sistema. La importancia de esta fase es fundamental ya que cualquier error o deficiencia en la codificación puede resultar en un software defectuoso.

Pruebas: En esta fase, se realizan pruebas para garantizar que el software funcione correctamente. Se prueban diferentes escenarios para asegurarse de que el software cumpla con los requisitos del usuario. La importancia de esta fase es fundamental ya que las pruebas son esenciales para garantizar la calidad del software y para detectar errores y problemas.

Mantenimiento: En esta fase, se realiza el mantenimiento del software después de su lanzamiento. Se realizan actualizaciones y correcciones de errores y se proporciona soporte técnico. La importancia de esta fase es fundamental ya que el mantenimiento adecuado del software garantiza que el software siga siendo eficiente y eficaz a lo largo del tiempo.

En resumen, cada fase del ciclo de vida del software es importante y tiene un papel específico en el desarrollo y mantenimiento de software de alta calidad. La comprensión y la realización adecuadas de cada fase son esenciales para garantizar que el software cumpla con los requisitos del usuario, sea eficiente y esté libre de errores.

El modelo en cascada, también conocido como el modelo de ciclo de vida en cascada, es un enfoque secuencial de desarrollo de software que implica una serie de fases bien definidas y secuenciales. Este modelo se utiliza ampliamente en la práctica debido a su simplicidad y facilidad de comprensión.

El modelo en cascada se compone de las siguientes fases:

Análisis de requisitos: En esta fase, se identifican y definen los requisitos del software.

Diseño: En esta fase, se diseña la arquitectura y la estructura del software.

Implementación: En esta fase, se codifica el software.

Pruebas: En esta fase, se prueban y validan los componentes del software.

Mantenimiento: En esta fase, se realiza el mantenimiento del software después de su lanzamiento.

Cada fase del modelo en cascada debe completarse antes de pasar a la siguiente fase. Este modelo se basa en la suposición de que los requisitos son conocidos y estables, lo que significa que cualquier cambio en los requisitos se debe hacer en la fase de análisis de requisitos y no en las etapas posteriores.

El modelo en cascada es ampliamente utilizado en la práctica debido a su simplicidad y facilidad de comprensión. Además, es fácil de administrar y se puede usar para proyectos pequeños y grandes. Sin embargo, este modelo tiene algunas limitaciones, como su falta de flexibilidad para cambios de requisitos, lo que puede llevar a retrasos en el desarrollo del software.

En conclusión, el modelo en cascada es un enfoque secuencial de desarrollo de software que se utiliza ampliamente en la práctica debido a su simplicidad y facilidad de comprensión. Sin embargo, debe tenerse en cuenta que este modelo puede tener limitaciones en términos de flexibilidad para cambios de requisitos.

El modelo en espiral es un modelo de desarrollo de software iterativo que combina los elementos del modelo en cascada y del modelo de prototipos. Este modelo se utiliza comúnmente para proyectos grandes y complejos donde los riesgos son altos y la incertidumbre es significativa.

El modelo en espiral se compone de las siguientes fases:

Planificación: En esta fase, se establecen los objetivos del proyecto, se definen las restricciones y se identifican los riesgos.

Análisis de riesgos: En esta fase, se identifican los riesgos asociados con el proyecto y se evalúan.

Desarrollo y validación: En esta fase, se desarrolla y valida una solución para el software.

Evaluación: En esta fase, se revisan los resultados y se evalúa el éxito del proyecto.

Cada fase del modelo en espiral se divide en cuatro cuadrantes, donde cada cuadrante representa una actividad diferente. El modelo en espiral utiliza un enfoque iterativo, lo que significa que cada fase se completa en varias iteraciones antes de avanzar a la siguiente fase.

El modelo en espiral se utiliza en la práctica porque permite la identificación temprana de riesgos y la mitigación de los mismos antes de que se conviertan en problemas mayores. Además, el modelo en espiral proporciona flexibilidad en términos de cambios de requisitos y especificaciones del software.

Sin embargo, el modelo en espiral también puede tener algunas limitaciones, como la complejidad del modelo y la necesidad de una planificación detallada y un control de riesgos constante, lo que puede ser costoso y llevar tiempo.

En resumen, el modelo en espiral es un enfoque iterativo y de gestión de riesgos para el desarrollo de software. Este modelo se utiliza comúnmente en proyectos grandes y complejos debido a su capacidad para identificar y mitigar los riesgos.

El modelo iterativo es un enfoque de desarrollo de software que implica la repetición de un ciclo de desarrollo corto varias veces hasta que se alcanza el objetivo final. Este modelo se utiliza comúnmente en proyectos donde los requisitos son inciertos o cambiantes, o cuando se requiere una mayor flexibilidad y adaptabilidad en el proceso de desarrollo.

El modelo iterativo se compone de las siguientes fases:

Planificación: En esta fase, se definen los objetivos, los requisitos y los entregables del proyecto.

Análisis y diseño: En esta fase, se desarrolla un diseño detallado del software y se identifican los riesgos.

Implementación: En esta fase, se implementa el software y se realizan pruebas iniciales.

Evaluación: En esta fase, se revisa y evalúa el software para determinar si cumple con los requisitos del proyecto.

Cada ciclo de desarrollo corto se compone de estas mismas fases, pero se centra en una funcionalidad o característica específica del software. Después de cada ciclo, se revisa y evalúa el progreso del proyecto y se toman decisiones sobre la continuación del proceso.

El modelo iterativo se utiliza en la práctica porque permite la adaptación continua del proceso de desarrollo y proporciona una mayor flexibilidad y adaptabilidad. Además, este modelo permite la entrega temprana de funcionalidades del software y una retroalimentación temprana del usuario.

Sin embargo, el modelo iterativo también puede tener algunas limitaciones, como la necesidad de una gestión cuidadosa del alcance del proyecto y el riesgo de un aumento en los costos y el tiempo de desarrollo si los ciclos de desarrollo se repiten con demasiada frecuencia.

En resumen, el modelo iterativo es un enfoque de desarrollo de software que implica la repetición de un ciclo de desarrollo corto varias veces hasta que se alcanza el objetivo final. Este modelo se utiliza comúnmente en proyectos donde los requisitos son inciertos o cambiantes, o cuando se requiere una mayor flexibilidad y adaptabilidad en el proceso de desarrollo.

7.3 Teorías que sustentan los modelos de ciclo de vida de software

Por ejemplo, el modelo en cascada se basa en la teoría de la gestión de proyectos, que establece que un proyecto debe ser planificado y ejecutado en fases secuenciales y que cada fase debe completarse antes de avanzar a la siguiente. Este modelo también se basa en la teoría de la ingeniería de sistemas, que se centra en la identificación y el análisis de los requisitos y la definición de los sistemas para cumplir con esos requisitos. Los trabajaos más relevantes son Gantt, H. L. (1910). Work, Wages, and Profits: Their Influence on the Cost of Production. The Engineering Magazine Company; Taylor, F. W. (1911). Principles of Scientific Management. Harper & Brothers y Fayol, H. (1916). General and Industrial Management. Pitman Publishing.

Por otro lado, el modelo iterativo se basa en la teoría de la retroalimentación continua y la mejora continua, donde los resultados y el aprendizaje se utilizan para mejorar continuamente el proceso y los resultados del proyecto. También se basa en la teoría de la gestión ágil de proyectos, que se centra en la colaboración, la entrega temprana de funcionalidades y la adaptación a los cambios.

Además, tanto el modelo en cascada como el modelo iterativo se basan en la teoría de la ingeniería de software, que se centra en la aplicación de principios y métodos de la ingeniería para el desarrollo de software.

En resumen, los modelos de ciclo de vida de software se basan en varias teorías, como la gestión de proyectos, la ingeniería de sistemas, la mejora continua y la gestión ágil de proyectos, entre otras. Estas teorías proporcionan un marco teórico para el desarrollo de software y ayudan a garantizar que el proceso sea sistemático, eficiente y efectivo.

7.4 Teorías más influyentes en el desarrollo de software y los modelos de ciclo de vida:

Teoría de la gestión de proyectos: Esta teoría se centra en la planificación, ejecución y control de proyectos. Algunos de los autores más influyentes en esta teoría son Henry Gantt, Frederick Winslow Taylor y Henri Fayol.

Teoría de la ingeniería de sistemas: Esta teoría se centra en la identificación y análisis de requisitos y la definición de sistemas para cumplir con esos requisitos. Algunos de los autores más influyentes en esta teoría son Barry Boehm, Eric Trist y C. West Churchman.

Teoría de la mejora continua: Esta teoría se centra en la retroalimentación continua y el aprendizaje para mejorar continuamente el proceso y los resultados del proyecto. Algunos de

los autores más influyentes en esta teoría son W. Edwards Deming, Joseph Juran y Kaoru Ishikawa.

Teoría de la gestión ágil de proyectos: Esta teoría se centra en la colaboración, la entrega temprana de funcionalidades y la adaptación a los cambios. Algunos de los autores más influyentes en esta teoría son Ken Schwaber, Jeff Sutherland y Alistair Cockburn.

En cuanto a los modelos de ciclo de vida de software, algunos de los autores y modelos más influyentes son:

Modelo en cascada: Este modelo fue propuesto por Winston Royce en 1970 y se basa en la secuencialidad de las fases del desarrollo de software.

Modelo en espiral: Este modelo fue propuesto por Barry Boehm en 1986 y se basa en la iteración y la retroalimentación continua.

Modelo iterativo: Este modelo se basa en la repetición de un ciclo de desarrollo corto varias veces hasta que se alcanza el objetivo final. Algunos de los autores más influyentes en esta teoría son James Martin, Edward Yourdon y Scott Ambler.

Es importante mencionar que hay muchos otros autores y teorías que han contribuido significativamente al desarrollo de software y a los modelos de ciclo de vida, y esta lista solo proporciona algunos ejemplos.

7.5 Metodologías de desarrollo de software.

Pressman (2014) define que: "Las metodologías de desarrollo de software son aproximaciones sistemáticas, disciplinadas y cuantificables al desarrollo, operación y mantenimiento del software".

Sommerville (2016) considera que: "Una metodología de desarrollo de software es un marco para la planificación, estructuración, diseño y control de procesos de desarrollo de software".

Boehm (1988) considera que: "Una metodología de desarrollo de software es un conjunto de actividades, acciones y tareas que se llevan a cabo durante el proceso de desarrollo de software y que se utilizan para planificar, diseñar, construir, probar y mantener el software".

Larman (2003) las define como: "Una metodología de desarrollo de software es un conjunto de técnicas, herramientas y prácticas que se utilizan para llevar a cabo el proceso de desarrollo de software".

En general, podemos decir que una metodología de desarrollo de software es un conjunto de prácticas, herramientas y técnicas que se utilizan para gestionar y controlar el proceso de desarrollo de software. Estas metodologías pueden variar en su grado de formalidad, enfoque y complejidad, y se utilizan para mejorar la eficiencia, calidad y productividad del proceso de desarrollo de software.

Existen varias metodologías de desarrollo de software que se utilizan en la práctica, cada una con sus propias características, ventajas y desventajas. A continuación, se describen algunas de las metodologías más populares:

Metodología de Desarrollo en Cascada: La metodología de desarrollo en cascada es una de las más antiguas y sencillas. Esta metodología se enfoca en completar cada etapa del ciclo de vida del software antes de avanzar a la siguiente. Esto significa que cada fase del proyecto se completa por completo antes de que comience la siguiente fase. Las características de esta metodología incluyen:

Estructura lineal y secuencial.

Cada fase depende del resultado de la fase anterior.

El cliente no tiene un rol activo en el proceso de desarrollo.

Ventajas:

Fácil de entender y de implementar.

Cada fase es completa antes de avanzar a la siguiente.

Cada etapa tiene un conjunto bien definido de entregables.

Desventajas:

El cliente no tiene un rol activo en el proceso de desarrollo.

Cambios en los requisitos pueden ser costosos y difíciles de implementar una vez que se ha completado la fase correspondiente.

El enfoque secuencial no se adapta bien a proyectos grandes y complejos.

Metodología de Desarrollo Iterativo: La metodología de desarrollo iterativo se enfoca en el desarrollo incremental del software. En esta metodología, el proyecto se divide en iteraciones o ciclos, y cada iteración es una versión funcional del software. En cada iteración se agregan más funcionalidades y características. Las características de esta metodología incluyen:

El software se desarrolla en iteraciones.

Las iteraciones sucesivas construyen y mejoran el producto.

Cada iteración es un mini proyecto.

Ventajas:

Permite una mayor interacción entre el equipo de desarrollo y el cliente.

Los cambios en los requisitos se pueden incorporar en las iteraciones subsiguientes.

El software funcional se entrega en etapas.

Desventajas:

La falta de una planificación detallada puede llevar a problemas de control de tiempo y presupuesto.

El proceso iterativo puede llevar a una falta de consistencia en la arquitectura del software.

No es adecuado para proyectos donde se requiere un enfoque lineal y secuencial.

Metodología Ágil: La metodología ágil se enfoca en la entrega de software en ciclos cortos e incrementales. Esta metodología es altamente colaborativa y se enfoca en responder

rápidamente a los cambios de los requisitos del cliente. Las características de esta metodología incluyen:

Ciclos cortos y frecuentes de entrega de software.

Equipo de desarrollo altamente colaborativo.

Se enfoca en responder rápidamente a los cambios de los requisitos del cliente.

Ventajas:

El cliente es un miembro activo del equipo de desarrollo.

El enfoque iterativo y colaborativo permite una respuesta más rápida a los cambios.

Mayor flexibilidad y adaptabilidad en el proceso de desarrollo.

Desventajas:

Puede haber falta de documentación y planificación detallada.

El control de calidad puede ser un desafío debido a la naturaleza iterativa.

No es adecuado para proyectos con requisitos muy específicos.

En conclusión, cada metodología de desarrollo de software tiene sus propias ventajas y desventajas, y la elección de la metodología adecuada

No hay una metodología única que sea la mejor para todos los proyectos de software, ya que cada proyecto tiene sus propias necesidades y características. Por lo tanto, la elección de la metodología adecuada depende de varios factores, como el tamaño del equipo de desarrollo, el alcance del proyecto, el presupuesto, la duración del proyecto, la experiencia del equipo de desarrollo y la complejidad del software a desarrollar.

A continuación, se presentan algunas recomendaciones generales sobre qué metodología de desarrollo de software utilizar según el tipo de proyecto:

Para proyectos pequeños y simples, la metodología ágil puede ser una buena opción, ya que permite una mayor flexibilidad en cuanto a la planificación y el desarrollo del software.

Para proyectos medianos y complejos, se puede considerar el uso de una metodología híbrida que combine elementos de diferentes metodologías, como Agile y Waterfall, para adaptarse a las necesidades específicas del proyecto.

Para proyectos grandes y complejos, que involucren a varios equipos y múltiples sistemas, se recomienda el uso de una metodología escalable, como SAFe o DAD, que proporcionan un marco de trabajo para la gestión de proyectos a gran escala.

Para proyectos críticos y de alta seguridad, se recomienda el uso de una metodología formal, como el enfoque V-model o el enfoque de desarrollo seguro de software (SDLC), que se centran en la calidad del software y la seguridad del sistema.

Es importante destacar que la elección de la metodología adecuada debe basarse en una evaluación cuidadosa de las necesidades y requisitos del proyecto, y debe ser revisada y

ajustada periódicamente durante el desarrollo del proyecto para garantizar que se sigan cumpliendo los objetivos y se logren los resultados deseados.

Las herramientas y técnicas utilizadas en cada metodología de desarrollo de software pueden variar según el enfoque específico de cada metodología. A continuación, se presentan algunas de las herramientas y técnicas más comunes utilizadas en algunas de las metodologías de desarrollo de software más populares:

Metodologías Ágiles:

Scrum: la metodología Scrum utiliza herramientas como el Product Backlog, Sprint Backlog, Sprint Review y Sprint Retrospective para gestionar el proceso de desarrollo.

Kanban: la metodología Kanban utiliza tableros Kanban y tarjetas para visualizar el flujo de trabajo y mejorar la gestión del proceso de desarrollo.

Extreme Programming (XP): la metodología XP utiliza técnicas como programación en parejas, integración continua y pruebas automatizadas para mejorar la calidad del software.

Metodologías de Cascada:

Diagrama de flujo de trabajo: esta técnica utiliza diagramas de flujo para visualizar el proceso de desarrollo de software.

Modelado de datos: el modelado de datos se utiliza para definir y estructurar la información y los datos del software.

Pruebas de sistema: las pruebas de sistema se realizan al final del proceso de desarrollo para probar y validar el software completo.

Metodologías Iterativas e Incrementales:

Desarrollo iterativo: en el desarrollo iterativo, se realizan varias iteraciones o ciclos de desarrollo que incluyen planificación, análisis, diseño, implementación y pruebas.

Desarrollo incremental: en el desarrollo incremental, el software se desarrolla en pequeñas partes funcionales, que se construyen y entregan incrementalmente.

Prototipado: el prototipado se utiliza para crear prototipos rápidos y simples del software para validar ideas y funcionalidades antes de su implementación.

Metodologías Lean:

Mejora continua: la metodología Lean se centra en la mejora continua del proceso de desarrollo mediante la eliminación de desperdicios y la optimización del flujo de trabajo.

Valor agregado: la metodología Lean se enfoca en agregar valor al software mediante la entrega de características y funcionalidades útiles al usuario final.

Es importante tener en cuenta que estas son solo algunas de las herramientas y técnicas utilizadas en cada metodología, y que la elección de herramientas y técnicas específicas puede variar según las necesidades y requisitos de cada proyecto de software.

7.6 Diseño de software

El diseño de software es la fase del proceso de desarrollo de software en la que se planifica y se crea la estructura detallada del software. Durante esta fase, los diseñadores de software toman las decisiones necesarias para definir cómo funcionará el software, cómo se organizarán los datos y cómo se interactuará con el usuario.

El diseño de software implica la creación de diagramas, modelos, y especificaciones detalladas que describen el funcionamiento del software. También se definen las interfaces de usuario, la arquitectura del software, la estructura de datos, y los algoritmos que se utilizarán en el software.

La fase de diseño de software es crucial para el éxito del proyecto de software. Un buen diseño puede permitir la creación de un software de alta calidad que cumpla con los requisitos del usuario y sea fácil de mantener y de actualizar. Por otro lado, un mal diseño puede llevar a problemas de calidad, rendimiento y mantenibilidad del software.

El diseño de software se realiza después de la fase de análisis de requisitos y antes de la fase de implementación. Durante esta fase, el equipo de desarrollo de software trabaja estrechamente con los stakeholders del proyecto, como los clientes y los usuarios finales, para asegurarse de que el diseño cumpla con sus necesidades y expectativas.

7.7 Principios de diseño de software

Los principios de diseño de software son un conjunto de directrices que se utilizan para crear un software bien diseñado, eficiente y fácil de mantener. Los cuatro principios de diseño de software más comunes son la modularidad, la abstracción, la cohesión y el acoplamiento.

Modularidad: La modularidad se refiere a la práctica de dividir el software en componentes más pequeños y manejables. Cada módulo debe ser independiente y estar diseñado para cumplir una tarea específica. La modularidad permite una mayor flexibilidad y reutilización de código, lo que a su vez facilita la integración de nuevos módulos o actualizaciones.

Abstracción: La abstracción se refiere a la práctica de identificar los elementos esenciales de un sistema y aislarlos de los detalles irrelevantes. La abstracción se utiliza para simplificar el diseño del software y reducir la complejidad. Esto hace que el software sea más fácil de entender, modificar y mantener.

Cohesión: La cohesión se refiere a la medida en que los componentes de un sistema están relacionados entre sí. Un sistema bien diseñado tendrá componentes altamente cohesivos, lo que significa que cada componente está diseñado para cumplir una tarea específica y relevante dentro del sistema. La cohesión permite una mayor facilidad de mantenimiento y una mayor flexibilidad en el diseño.

Acoplamiento: El acoplamiento se refiere a la medida en que los componentes de un sistema dependen entre sí. Un sistema con bajo acoplamiento tendrá componentes independientes que pueden ser modificados sin afectar el funcionamiento del resto del sistema. El acoplamiento bajo también permite una mayor flexibilidad en el diseño y reduce el riesgo de errores y problemas de rendimiento.

La aplicación de estos principios de diseño de software puede ayudar a los desarrolladores a crear software bien estructurado, fácil de mantener y que cumpla con los requisitos del usuario. Al aplicar estos principios, los desarrolladores pueden asegurar que su software sea escalable, robusto y de alta calidad.

7.8 Cómo aplicar los principios para diseñar software de alta calidad.

Para aplicar los principios de diseño de software en la práctica, se pueden seguir los siguientes pasos:

Identificar los requisitos: Es importante conocer los requisitos del usuario para determinar qué funcionalidades debe tener el software.

Dividir el software en módulos: Una vez identificados los requisitos, se deben dividir en módulos más pequeños y manejables. Cada módulo debe estar diseñado para cumplir una tarea específica y ser independiente.

Diseñar la arquitectura: La arquitectura del software debe ser coherente con los módulos identificados. Es importante tener en cuenta la cohesión y el acoplamiento de los módulos para asegurar que el software sea fácil de mantener y escalar.

Aplicar la abstracción: Identificar los elementos esenciales del software y aislarlos de los detalles irrelevantes. Por ejemplo, se pueden utilizar patrones de diseño para abstraer las funcionalidades comunes y simplificar el diseño.

Reutilización de código: Se deben utilizar módulos existentes siempre que sea posible para reducir la complejidad y el tiempo de desarrollo.

Pruebas y verificación: Es importante probar cada módulo por separado y el software en su conjunto para garantizar que cumpla con los requisitos del usuario.

7.9 Mantenimiento del software

El mantenimiento del software es crucial para mantenerlo actualizado y funcionando correctamente. La modularidad y la cohesión del diseño facilitan el mantenimiento y la actualización del software.

Al seguir estos pasos y aplicar los principios de diseño de software, se puede crear software de alta calidad que sea fácil de mantener y escalable. La modularidad, la abstracción, la cohesión y el acoplamiento son principios fundamentales que deben tenerse en cuenta en todo momento durante el proceso de diseño del software.

Aquí proporcionamos algunos ejemplos de diseños de software efectivos y por qué se pueden considerar buenos ejemplos:

El sistema operativo Unix: Este sistema operativo es considerado uno de los mejores ejemplos de diseño de software debido a su enfoque modular y su diseño de bajo acoplamiento. Unix utiliza una arquitectura de capas para separar las funcionalidades en módulos independientes.

Cada capa tiene una interfaz claramente definida para reducir el acoplamiento y mejorar la cohesión.

El framework de desarrollo web Ruby on Rails: Este framework es conocido por su diseño simple y elegante que se centra en la convención sobre la configuración. Ruby on Rails utiliza el patrón de diseño Modelo-Vista-Controlador (MVC) para separar las funcionalidades y simplificar el desarrollo. El diseño de Ruby on Rails hace que sea fácil crear aplicaciones web escalables y fáciles de mantener.

La aplicación de procesamiento de texto Microsoft Word: A pesar de su complejidad, Microsoft Word es un buen ejemplo de diseño de software debido a su enfoque modular y su diseño orientado a objetos. La aplicación se divide en módulos separados que se centran en tareas específicas, como el procesamiento de texto, la manipulación de gráficos y la corrección ortográfica. Cada módulo está diseñado para ser independiente y se comunica con otros módulos a través de una interfaz claramente definida.

El juego Minecraft: Minecraft es un juego popular que es conocido por su diseño escalable y flexible. El juego utiliza un enfoque modular para crear objetos y paisajes que se pueden modificar y construir en tiempo real. El diseño modular de Minecraft permite a los jugadores crear y personalizar su propia experiencia de juego, lo que ha contribuido en gran medida a su popularidad.

Estos son solo algunos ejemplos de diseños de software efectivos. Cada uno se centra en la modularidad, la abstracción, la cohesión y el acoplamiento para crear un software fácil de mantener, escalable y efectivo.

7.10 Herramientas de ingeniería de software

A continuación, se presentan algunas de las herramientas de software más comunes utilizadas en el desarrollo de software:

Entornos de desarrollo integrados (IDE): Son aplicaciones que proporcionan un entorno completo para escribir, probar y depurar código. Ejemplos de IDE populares incluyen Eclipse, Visual Studio y IntelliJ IDEA.

Sistemas de control de versiones: Son herramientas que permiten a los desarrolladores rastrear los cambios en el código fuente a lo largo del tiempo. Ejemplos de sistemas de control de versiones populares incluyen Git, SVN y Mercurial.

Herramientas de automatización de pruebas: Son herramientas que ayudan a los desarrolladores a probar automáticamente el código. Ejemplos de herramientas de automatización de pruebas incluyen Selenium, JUnit y TestNG.

Gestores de dependencias: Son herramientas que ayudan a los desarrolladores a administrar las dependencias del software. Ejemplos de gestores de dependencias populares incluyen Maven, Gradle y npm.

Herramientas de integración continua: Son herramientas que automatizan el proceso de construcción y prueba del software en un entorno controlado y aislado. Ejemplos de herramientas de integración continua incluyen Jenkins, Travis CI y CircleCI.

Herramientas de seguimiento de problemas: Son herramientas que ayudan a los desarrolladores a rastrear y resolver problemas relacionados con el software. Ejemplos de herramientas de seguimiento de problemas populares incluyen JIRA, Bugzilla y Redmine.

Herramientas de colaboración: Son herramientas que ayudan a los desarrolladores a colaborar entre sí en el desarrollo de software. Ejemplos de herramientas de colaboración incluyen Slack, Microsoft Teams y Google Workspace.

Estas son algunas de las herramientas de software comunes utilizadas en el desarrollo de software. Cada herramienta tiene su propio conjunto de características y beneficios, y la elección de una herramienta dependerá de las necesidades específicas del proyecto de desarrollo de software.

Los editores de código son herramientas de software diseñadas para facilitar la escritura, edición y organización de código fuente. A diferencia de los IDE (Entornos de Desarrollo Integrados), que incluyen una serie de herramientas integradas para la construcción, prueba y depuración de software, los editores de código se centran principalmente en la edición de código fuente.

Los editores de código suelen ser aplicaciones ligeras y rápidas que están diseñadas para trabajar con una amplia variedad de lenguajes de programación. A menudo, los editores de código tienen características como la resaltado de sintaxis, la auto-completado, la corrección de errores, y la capacidad de manejar múltiples archivos a la vez.

Algunos editores de código populares incluyen Visual Studio Code, Sublime Text, Atom, Brackets y Notepad++. Estos editores son gratuitos y de código abierto, y ofrecen una amplia gama de extensiones y complementos para mejorar su funcionalidad y adaptarse a las necesidades de cada usuario.

Los editores de código son herramientas esenciales para cualquier programador ya que les permiten trabajar de manera más eficiente y productiva en el proceso de creación de software.

Los editores de código se utilizan para escribir, editar y organizar el código fuente de un programa o aplicación. A continuación, se muestran los pasos básicos para utilizar un editor de código:

Descarga e instala un editor de código: Puedes descargar e instalar un editor de código de forma gratuita desde la página web del editor de tu elección. Una vez que lo hayas descargado, sigue las instrucciones para instalarlo en tu equipo.

Abre el editor de código: Una vez instalado, abre el editor de código. Verás una pantalla en blanco que te permitirá empezar a escribir código.

Escribe el código fuente: Escribe el código fuente de tu programa o aplicación en el editor de código. A medida que escribas, el editor te proporcionará características como la resaltado de sintaxis, la auto-completado y la corrección de errores para ayudarte a escribir un código limpio y libre de errores.

Guarda tu archivo: Una vez que hayas terminado de escribir el código, guarda tu archivo en tu equipo. El archivo se guardará con una extensión de archivo específica para el lenguaje de programación que estás usando.

Compila y ejecuta el código: Después de guardar el archivo, puedes compilar y ejecutar el código utilizando otras herramientas de software, como un compilador o un intérprete de lenguaje de programación.

Repite los pasos 3 a 5: Repite los pasos 3 a 5 para escribir y editar tu código fuente hasta que hayas completado tu programa o aplicación.

Los editores de código son herramientas esenciales para los programadores que les permiten escribir, editar y organizar el código fuente de sus programas y aplicaciones de manera eficiente y productiva.

A continuación, se muestran algunas ventajas y desventajas de utilizar editores de código:

Ventajas:

Eficiencia: Los editores de código están diseñados para hacer que la escritura y edición de código sea más rápida y eficiente. Algunas características como la resaltado de sintaxis, la auto-completado y la corrección de errores pueden ahorrar tiempo y mejorar la productividad.

Flexibilidad: Los editores de código son muy flexibles, ya que pueden trabajar con una amplia variedad de lenguajes de programación. Esto significa que los programadores pueden usar un solo editor para escribir código en varios lenguajes.

Personalización: Los editores de código ofrecen una amplia gama de extensiones y complementos que pueden personalizarse para adaptarse a las necesidades de cada usuario. Los programadores pueden agregar o quitar características según sus preferencias.

Desventajas:

Falta de herramientas integradas: A diferencia de los IDE, los editores de código no ofrecen herramientas integradas para la construcción, prueba y depuración de software. Los programadores pueden necesitar usar otras herramientas de software para completar estas tareas.

Curva de aprendizaje: A pesar de que los editores de código son relativamente fáciles de usar, algunos de ellos tienen una curva de aprendizaje más pronunciada. Los usuarios pueden necesitar invertir tiempo para aprender a usar todas las características y extensiones de un editor de código.

Limitaciones en proyectos grandes: Para proyectos grandes y complejos, los editores de código pueden no ser suficientes. Los IDEs ofrecen más herramientas y características para manejar proyectos grandes y complejos.

En resumen, los editores de código ofrecen muchas ventajas para los programadores, como eficiencia, flexibilidad y personalización. Sin embargo, también tienen algunas desventajas, como la falta de herramientas integradas y la curva de aprendizaje. Los programadores deben considerar cuidadosamente las necesidades de su proyecto antes de elegir un editor de código.

Un compilador es un programa que traduce el código fuente escrito en un lenguaje de programación de alto nivel a un lenguaje de máquina (código objeto) que una computadora puede ejecutar. A continuación, se describen algunos aspectos clave de los compiladores:

Función principal: El trabajo principal de un compilador es traducir el código fuente de un programa escrito en un lenguaje de programación de alto nivel (como C++, Java o Python) a código objeto (código de máquina) que la computadora puede ejecutar. El compilador realiza un análisis sintáctico y semántico del código fuente para detectar cualquier error y produce un archivo ejecutable.

Fases del proceso de compilación: El proceso de compilación consta de varias fases que incluyen análisis léxico, análisis sintáctico, análisis semántico, generación de código intermedio, optimización de código y generación de código de máquina. Cada fase del proceso tiene un propósito específico y es realizada por diferentes componentes del compilador.

Tipos de compiladores: Hay varios tipos de compiladores, como compiladores de una sola pasada, compiladores de varias pasadas, compiladores JIT (Just-in-time) y compiladores de análisis sintáctico dirigido por sintaxis. Cada tipo de compilador tiene sus propias características y ventajas.

Uso de los compiladores: Los compiladores se utilizan en el proceso de desarrollo de software para traducir el código fuente de un programa a un formato que pueda ser ejecutado por una computadora. También se utilizan en la creación de sistemas operativos, compiladores de lenguaje, y en la creación de otros programas de software.

En resumen, un compilador es un programa que traduce el código fuente de un programa escrito en un lenguaje de programación de alto nivel a código objeto que una computadora puede ejecutar. Los compiladores realizan una serie de fases de procesamiento del código fuente y hay varios tipos de compiladores disponibles para los programadores. Los compiladores son esenciales para el desarrollo de software y la creación de otros programas.

Los compiladores son de gran importancia en el proceso de desarrollo de software por varias razones:

Optimización de código: Los compiladores son capaces de optimizar el código generado, mejorando su eficiencia y rendimiento. Al analizar el código fuente, el compilador puede identificar oportunidades para optimizar el código, como eliminar código redundante, reorganizar el código para aprovechar mejor el procesador y reducir el número de operaciones realizadas.

Eficiencia: Los compiladores pueden generar código objeto altamente eficiente para una plataforma específica. Esto significa que el programa compilado se ejecutará más rápido y utilizará menos recursos que si se escribiera en lenguaje de bajo nivel directamente.

Portabilidad: Los compiladores permiten a los programadores escribir un programa en un lenguaje de programación de alto nivel y luego compilarlo para diferentes plataformas. Esto hace que el programa sea más portátil y fácil de distribuir.

Corrección de errores: Los compiladores realizan un análisis sintáctico y semántico del código fuente para detectar errores antes de la compilación. Esto ayuda a los programadores a encontrar y corregir errores de código más rápidamente.

Seguridad: Los compiladores pueden ayudar a mejorar la seguridad de un programa. Los compiladores modernos pueden detectar y prevenir vulnerabilidades comunes, como desbordamientos de búfer y ataques de inyección de código.

En resumen, los compiladores son importantes en el proceso de desarrollo de software porque mejoran la eficiencia del código, hacen que el programa sea más portátil, ayudan a encontrar y corregir errores y pueden mejorar la seguridad del programa.

Ventajas de los compiladores:

Mayor eficiencia: El código compilado se ejecuta más rápido y consume menos recursos que el código interpretado.

Optimización de código: Los compiladores pueden optimizar el código fuente para mejorar su rendimiento y eficiencia.

Portabilidad: Los programas compilados pueden ejecutarse en diferentes plataformas sin necesidad de modificar el código fuente.

Mayor seguridad: Los compiladores modernos pueden detectar y prevenir vulnerabilidades comunes, mejorando la seguridad del programa.

Mayor control: El proceso de compilación ofrece un mayor control sobre el programa y su comportamiento, lo que puede ser útil en situaciones en las que se requiere un mayor grado de precisión.

Desventajas de los compiladores:

Curva de aprendizaje: Aprender a utilizar un compilador puede llevar tiempo y esfuerzo.

Dificultad de depuración: El proceso de depuración puede ser más complicado con el código compilado que con el código interpretado.

Falta de flexibilidad: Los programas compilados no son tan flexibles como los programas interpretados, ya que el código compilado no puede modificarse en tiempo de ejecución.

Dependencia del hardware: Los programas compilados están diseñados para un hardware específico, lo que significa que puede ser necesario compilar el código fuente para diferentes plataformas.

Mayor complejidad: El proceso de compilación puede ser más complejo que el proceso de interpretación, lo que puede resultar en una curva de aprendizaje más empinada y en una mayor cantidad de errores.

Los depuradores (o debuggers en inglés) son herramientas de software que se utilizan para identificar y solucionar problemas en el código fuente de un programa. Estas herramientas permiten a los desarrolladores examinar el estado interno de un programa en tiempo de ejecución y detener la ejecución del programa en puntos específicos para examinar el estado de las variables y otros datos.

Los depuradores generalmente ofrecen una variedad de funciones, como establecer puntos de interrupción, examinar el valor de las variables, ejecutar el código línea por línea, examinar la pila de llamadas, entre otras.

Además de estas funciones básicas, algunos depuradores también ofrecen características avanzadas, como seguimiento de memoria, visualización de hilos de ejecución, análisis de código en tiempo real, entre otras.

En resumen, los depuradores son herramientas esenciales en el proceso de desarrollo de software, ya que permiten a los desarrolladores detectar y solucionar errores y problemas en el código fuente de un programa. Al proporcionar una vista detallada del estado interno del programa en tiempo de ejecución, los depuradores ayudan a los desarrolladores a comprender el comportamiento del programa y a corregir errores de forma más eficiente.

Los depuradores son una herramienta fundamental en el proceso de desarrollo de software y tienen una gran importancia por varias razones:

Identificación de errores: Los depuradores son esenciales para identificar y solucionar errores en el código fuente de un programa. Al permitir a los desarrolladores examinar el estado interno del programa en tiempo de ejecución, los depuradores facilitan la identificación de errores y la corrección de problemas.

Ahorro de tiempo: Los depuradores pueden ayudar a los desarrolladores a ahorrar tiempo al permitirles identificar y solucionar problemas de forma más eficiente. En lugar de tener que buscar manualmente el origen de un problema, los depuradores pueden señalar el problema específico y permitir que los desarrolladores corrijan el problema más rápidamente.

Mayor comprensión del código: Los depuradores permiten a los desarrolladores examinar el estado interno del programa y comprender mejor cómo funciona el código. Esto puede ayudar a los desarrolladores a mejorar la calidad del código y a hacer que el programa sea más eficiente y robusto.

Corrección de errores en producción: Los depuradores también son útiles en la identificación y solución de errores en producción. Al permitir a los desarrolladores examinar el estado interno del programa en tiempo real, los depuradores pueden ayudar a corregir problemas críticos que ocurren en el entorno de producción.

Depuración de sistemas embebidos: Los depuradores son particularmente útiles en la depuración de sistemas embebidos, donde es difícil o imposible depurar el sistema sin herramientas especiales. Los depuradores pueden ayudar a identificar y solucionar problemas en sistemas embebidos y mejorar su eficiencia y rendimiento.

En resumen, los depuradores son herramientas esenciales en el proceso de desarrollo de software y son importantes porque permiten a los desarrolladores identificar y solucionar errores, ahorrar tiempo, comprender mejor el código, corregir errores en producción y depurar sistemas embebidos.

Las ventajas y desventajas de los depuradores son las siguientes:

Ventajas:

Identificación de errores: Los depuradores permiten identificar y corregir errores de forma más eficiente y precisa que otras herramientas de análisis de código.

Mejora la calidad del código: Los depuradores ayudan a los desarrolladores a comprender mejor el código y a detectar problemas y malas prácticas en el mismo. Al corregir estos problemas, se puede mejorar la calidad del código.

Ahorro de tiempo: Los depuradores pueden ahorrar mucho tiempo a los desarrolladores, especialmente cuando se trata de encontrar y corregir errores complejos.

Depuración en tiempo real: Los depuradores permiten a los desarrolladores examinar el estado interno del programa en tiempo real y en detalle, lo que facilita la solución de problemas.

Identificación de problemas en producción: Los depuradores también pueden ser útiles en la identificación de problemas en entornos de producción.

Desventajas:

Curva de aprendizaje: Los depuradores pueden ser herramientas complicadas de aprender y dominar, especialmente para los principiantes.

Ralentización del programa: Los depuradores pueden ralentizar el programa durante la ejecución, lo que puede ser especialmente problemático en programas que requieren un alto rendimiento.

Limitaciones del depurador: Los depuradores no siempre pueden detectar ciertos tipos de errores o problemas, especialmente aquellos que se producen en situaciones de producción.

Problemas con sistemas embebidos: Los depuradores pueden no ser efectivos en sistemas embebidos o en programas que se ejecutan en dispositivos remotos o limitados en recursos.

En general, los depuradores son herramientas esenciales para el desarrollo de software, pero tienen sus ventajas y desventajas. Aunque pueden ser complicados de aprender y usar, pueden mejorar significativamente la calidad y la eficiencia del código, y son una herramienta importante para identificar y corregir errores en el código.

Los gestores de versiones son herramientas que se utilizan en el desarrollo de software para gestionar y controlar los cambios en el código fuente. Su función principal es ayudar a los desarrolladores a mantener un historial completo de todas las versiones del código y permitirles trabajar en equipo de manera más eficiente.

Existen dos tipos principales de gestores de versiones:

Sistemas de control de versiones centralizados (CVCS): Estos sistemas utilizan un servidor centralizado para almacenar el código fuente y mantener un registro de los cambios. Cuando un desarrollador desea realizar cambios en el código, debe descargar una copia del mismo del servidor central y trabajar en su propia copia local. Cuando se completan los cambios, el desarrollador lo carga de vuelta al servidor central, lo que permite que otros miembros del equipo descarguen la versión actualizada.

Sistemas de control de versiones distribuidos (DVCS): Estos sistemas permiten que cada miembro del equipo tenga una copia completa del repositorio de código en su propia máquina.

Los desarrolladores pueden trabajar en sus copias locales y luego enviar sus cambios a otros miembros del equipo mediante la sincronización con los demás repositorios. Esto hace que los sistemas DVCS sean más flexibles y permiten que los desarrolladores trabajen sin conexión a Internet.

Ambos tipos de gestores de versiones proporcionan una serie de características útiles para los desarrolladores, incluyendo:

Control de versiones: Los gestores de versiones permiten a los desarrolladores mantener un historial completo de todas las versiones del código, lo que facilita la identificación de problemas y la restauración de versiones anteriores.

Colaboración: Los gestores de versiones permiten que varios miembros del equipo trabajen en el mismo código fuente sin temor a perder trabajo. Cada desarrollador puede trabajar en su propia rama de desarrollo y, cuando esté listo, puede fusionar sus cambios con el código principal.

Etiquetado y ramificación: Los gestores de versiones permiten etiquetar versiones específicas del código para marcar hitos importantes y ramificar el código para realizar cambios importantes sin afectar el código principal.

Anotaciones y comentarios: Los gestores de versiones permiten a los desarrolladores añadir comentarios y anotaciones a los cambios de código, lo que ayuda a mantener una documentación clara y a comprender mejor los cambios realizados en el código.

En resumen, los gestores de versiones son herramientas esenciales para el desarrollo de software, que permiten a los desarrolladores mantener un historial completo de todas las versiones del código, colaborar de manera efectiva y trabajar de manera más eficiente en equipo.

Los gestores de versiones son herramientas esenciales para el desarrollo de software, ya que proporcionan una forma eficiente y segura de gestionar y controlar el código fuente. Algunas de las principales razones por las que son importantes son:

Control de versiones: Los gestores de versiones permiten a los desarrolladores mantener un historial completo de todas las versiones del código, lo que facilita la identificación de problemas y la restauración de versiones anteriores.

Colaboración: Los gestores de versiones permiten que varios miembros del equipo trabajen en el mismo código fuente sin temor a perder trabajo. Cada desarrollador puede trabajar en su propia rama de desarrollo y, cuando esté listo, puede fusionar sus cambios con el código principal.

Etiquetado y ramificación: Los gestores de versiones permiten etiquetar versiones específicas del código para marcar hitos importantes y ramificar el código para realizar cambios importantes sin afectar el código principal.

Seguridad: Los gestores de versiones proporcionan una capa adicional de seguridad al permitir que los desarrolladores realicen copias de seguridad regulares del código fuente y recuperen versiones anteriores en caso de errores o fallos.

Documentación: Los gestores de versiones permiten a los desarrolladores añadir comentarios y anotaciones a los cambios de código, lo que ayuda a mantener una documentación clara y a comprender mejor los cambios realizados en el código.

En general, los gestores de versiones ayudan a mejorar la eficiencia y la calidad del desarrollo de software, lo que es crucial en un entorno de desarrollo en constante evolución y cambio.

Los sistemas de control de versiones centralizados (CVCS) tienen ventajas y desventajas:

Ventajas:

Facilidad de uso: Los sistemas de control de versiones centralizados son muy fáciles de usar y requieren poco o ningún conocimiento técnico especializado para comenzar a usarlos.

Control centralizado: Todos los archivos y cambios están almacenados en un solo lugar, lo que facilita el control y la gestión centralizada del código fuente.

Seguridad: Los sistemas de control de versiones centralizados suelen contar con medidas de seguridad más avanzadas que los sistemas de control de versiones distribuidos, ya que los archivos están almacenados en un servidor centralizado y se pueden implementar medidas de seguridad en el nivel del servidor.

Facilidad de colaboración: Los sistemas de control de versiones centralizados son ideales para proyectos pequeños o medianos que involucran a un número limitado de desarrolladores, ya que facilitan la colaboración y la sincronización del código fuente.

Desventajas:

Fallos en el servidor: Si el servidor centralizado falla, se puede perder todo el historial de versiones y el código fuente. Esto puede ser un problema crítico para proyectos grandes y de misión crítica.

Dependencia del servidor: Los sistemas de control de versiones centralizados requieren que los desarrolladores tengan acceso constante al servidor centralizado para realizar cambios en el código fuente. Si hay problemas de conectividad, esto puede ser un obstáculo importante para el trabajo del equipo de desarrollo.

Cuello de botella: Con los sistemas de control de versiones centralizados, todas las operaciones deben realizarse a través del servidor centralizado, lo que puede generar cuellos de botella en la red y reducir la velocidad y eficiencia de las operaciones.

Menos flexibilidad: Los sistemas de control de versiones centralizados ofrecen menos flexibilidad en comparación con los sistemas de control de versiones distribuidos, ya que las operaciones están limitadas por las capacidades del servidor centralizado y la arquitectura del sistema.

Los sistemas de control de versiones distribuidos (DVCS) tienen ventajas y desventajas:

Ventajas:

Sin dependencia del servidor: Los sistemas de control de versiones distribuidos no requieren acceso constante a un servidor centralizado, lo que significa que los desarrolladores pueden

trabajar en el código fuente incluso cuando no hay conectividad a Internet o cuando el servidor centralizado está inaccesible.

Flexibilidad: Los sistemas de control de versiones distribuidos son muy flexibles y pueden adaptarse a diferentes arquitecturas de proyectos, desde proyectos pequeños y simples hasta proyectos grandes y complejos.

Gestión de ramas: Los sistemas de control de versiones distribuidos ofrecen una gestión de ramas mucho más avanzada que los sistemas centralizados, lo que facilita la colaboración y permite a los desarrolladores trabajar en diferentes ramas del proyecto al mismo tiempo.

Copias locales completas: Los sistemas de control de versiones distribuidos almacenan una copia completa del repositorio de código en cada equipo de desarrollo, lo que significa que los desarrolladores pueden trabajar con una copia completa del repositorio de código sin necesidad de acceder al servidor centralizado.

Desventajas:

Mayor complejidad: Los sistemas de control de versiones distribuidos pueden ser más complicados de configurar y utilizar que los sistemas centralizados, especialmente para los desarrolladores sin experiencia en el uso de herramientas de control de versiones.

Mayor demanda de recursos: Los sistemas de control de versiones distribuidos requieren más recursos de hardware y software que los sistemas centralizados, ya que cada equipo de desarrollo debe almacenar una copia completa del repositorio de código.

Posibles problemas de seguridad: Debido a que cada equipo de desarrollo almacena una copia completa del repositorio de código, los sistemas de control de versiones distribuidos pueden presentar más riesgos de seguridad que los sistemas centralizados si no se implementan adecuadas medidas de seguridad.

Mayor complejidad de gestión de ramas: Aunque los sistemas de control de versiones distribuidos ofrecen una gestión avanzada de ramas, esta flexibilidad puede hacer que la gestión de ramas sea más compleja que en los sistemas centralizados, especialmente para equipos grandes y proyectos complejos.

7.11 Cómo se utilizan las herramientas en la práctica

Las herramientas de software que se utilizan en el desarrollo de software se utilizan en diferentes etapas del proceso de desarrollo de software. A continuación, se describen algunos ejemplos de cómo se utilizan estas herramientas en la práctica:

Editores de código: Los editores de código se utilizan para escribir y editar código fuente. Los desarrolladores pueden usar diferentes editores de código, según sus preferencias y las necesidades del proyecto. Por ejemplo, algunos editores de código populares incluyen Visual Studio Code, Sublime Text y Atom. Los desarrolladores pueden personalizar estos editores de código con diferentes complementos y extensiones para mejorar la productividad y la eficiencia.

Compiladores: Los compiladores se utilizan para convertir el código fuente en un formato que la máquina pueda ejecutar. Los desarrolladores pueden utilizar diferentes compiladores según el lenguaje de programación que estén utilizando. Por ejemplo, si están programando en C++, pueden utilizar el compilador GNU GCC.

Depuradores: Los depuradores se utilizan para encontrar y corregir errores en el código. Los desarrolladores pueden utilizar diferentes depuradores para diferentes lenguajes de programación. Por ejemplo, los desarrolladores que trabajan con Java pueden utilizar el depurador integrado en la plataforma Java Development Kit (JDK) o el depurador Eclipse.

Sistemas de control de versiones: Los sistemas de control de versiones se utilizan para mantener un historial de los cambios realizados en el código fuente. Los desarrolladores pueden utilizar diferentes sistemas de control de versiones según las necesidades del proyecto. Por ejemplo, pueden utilizar Git para proyectos distribuidos o SVN para proyectos centralizados.

En general, estas herramientas se utilizan en conjunto para facilitar el desarrollo de software y mejorar la eficiencia del proceso de desarrollo. Por ejemplo, los desarrolladores pueden utilizar un editor de código para escribir y editar el código fuente, un compilador para convertir el código en un formato ejecutable, un depurador para encontrar y corregir errores y un sistema de control de versiones para mantener un historial de los cambios realizados en el código. Al utilizar estas herramientas de manera efectiva, los desarrolladores pueden aumentar su productividad y mejorar la calidad del software que están desarrollando.

un ejemplo práctico de cómo se utilizan estas herramientas en el proceso de desarrollo de software:

Supongamos que un equipo de desarrollo está trabajando en un proyecto de software utilizando el lenguaje de programación Python. En este caso, podrían utilizar las siguientes herramientas:

Editores de código: Los desarrolladores podrían utilizar un editor de código como PyCharm para escribir y editar el código fuente.

Compiladores: Como Python es un lenguaje interpretado, no requiere de un compilador para convertir el código fuente en un formato ejecutable. Sin embargo, los desarrolladores podrían utilizar herramientas como pip para instalar y gestionar las dependencias del proyecto.

Depuradores: Para encontrar y corregir errores en el código, los desarrolladores podrían utilizar un depurador como el integrado en PyCharm o el depurador de Python pdb.

Sistemas de control de versiones: Para mantener un historial de los cambios realizados en el código fuente, los desarrolladores podrían utilizar un sistema de control de versiones como Git. Podrían crear un repositorio Git para el proyecto y utilizar herramientas como GitHub o GitLab para gestionar y colaborar en el desarrollo del proyecto.

En este ejemplo, las herramientas de software se utilizan en conjunto para facilitar el proceso de desarrollo de software. Los desarrolladores utilizan un editor de código para escribir y editar el código fuente, una herramienta de gestión de dependencias para instalar y gestionar las bibliotecas requeridas, un depurador para encontrar y corregir errores y un sistema de control de versiones para mantener un historial de los cambios realizados en el código. Al utilizar estas

herramientas de manera efectiva, los desarrolladores pueden trabajar de manera más eficiente y colaborar mejor en el desarrollo del proyecto.

7.12 Pruebas de software

Las pruebas de software son el proceso de verificar y validar el software para garantizar que cumple con los requisitos establecidos y que funciona correctamente. El objetivo principal de las pruebas de software es detectar errores, defectos o problemas en el software y asegurarse de que se corrijan antes de que el software sea lanzado al mercado o utilizado por los usuarios.

Las pruebas de software pueden incluir diferentes tipos de pruebas, como pruebas funcionales, pruebas de rendimiento, pruebas de seguridad y pruebas de usabilidad. Las pruebas pueden ser manuales o automatizadas, dependiendo de los recursos disponibles y de las necesidades del proyecto. También pueden ser realizadas por equipos internos de pruebas de software o por terceros especializados en pruebas de software.

En general, las pruebas de software son una parte crítica del proceso de desarrollo de software, ya que ayudan a garantizar la calidad del software y reducir los riesgos asociados con su uso.

Las pruebas de software son el proceso de verificar y validar el software para garantizar que cumple con los requisitos establecidos y que funciona correctamente. El objetivo principal de las pruebas de software es detectar errores, defectos o problemas en el software y asegurarse de que se corrijan antes de que el software sea lanzado al mercado o utilizado por los usuarios.

Las pruebas de software pueden incluir diferentes tipos de pruebas, como pruebas funcionales, pruebas de rendimiento, pruebas de seguridad y pruebas de usabilidad. Las pruebas pueden ser manuales o automatizadas, dependiendo de los recursos disponibles y de las necesidades del proyecto. También pueden ser realizadas por equipos internos de pruebas de software o por terceros especializados en pruebas de software.

En general, las pruebas de software son una parte crítica del proceso de desarrollo de software, ya que ayudan a garantizar la calidad del software y reducir los riesgos asociados con su uso.

Existen diferentes tipos de pruebas de software, cada una enfocada en verificar un aspecto específico del software. Algunos de los tipos más comunes son:

Pruebas unitarias: Son pruebas realizadas a nivel de componente individual del software, como una función o un módulo. Estas pruebas se realizan para comprobar que cada componente funciona correctamente según su diseño.

Las pruebas unitarias se realizan en el código fuente de un componente individual del software, generalmente escrito por el desarrollador. La idea es probar el código en aislamiento, verificando que el resultado de la ejecución sea el esperado. Las mejores prácticas en pruebas unitarias incluyen escribir pruebas que cubran todos los caminos posibles del código, verificar las excepciones y errores, y mantener las pruebas actualizadas con los cambios del código.

Pruebas de integración: Una vez que los componentes individuales se han probado y verificado, es necesario comprobar cómo se integran y funcionan juntos. Las pruebas de integración buscan detectar problemas en la interacción entre los diferentes componentes.

Las pruebas de integración se realizan después de las pruebas unitarias, para verificar que los componentes individuales se integren correctamente y funcionen juntos como un sistema. Las mejores prácticas en pruebas de integración incluyen identificar los componentes críticos para la integración, probar en entornos similares al entorno de producción, y documentar los resultados de las pruebas.

Pruebas de sistema: Las pruebas de sistema se realizan a nivel de todo el sistema o aplicación. Estas pruebas se realizan para verificar que el software funciona correctamente en su entorno y para comprobar que cumple con los requisitos establecidos.

Las pruebas de sistema se realizan en el sistema completo y se enfocan en verificar que el software cumpla con los requisitos funcionales y no funcionales. Las mejores prácticas en pruebas de sistema incluyen probar en entornos similares al entorno de producción, utilizar conjuntos de datos realistas, y documentar los resultados de las pruebas.

Pruebas de aceptación: Las pruebas de aceptación son las pruebas finales que se realizan antes de que el software sea entregado al cliente o al usuario final. Estas pruebas se realizan para comprobar que el software cumple con los requisitos del cliente y que funciona correctamente en el entorno de producción.

Las pruebas de aceptación se realizan para verificar que el software cumpla con los requisitos del cliente y esté listo para su entrega. Estas pruebas suelen ser realizadas por el cliente o el usuario final. Las mejores prácticas en pruebas de aceptación incluyen involucrar al cliente desde el principio del proyecto, crear casos de prueba realistas basados en los requisitos del cliente, y documentar los resultados de las pruebas.

Además de estos tipos básicos de pruebas, también existen otros tipos de pruebas como las pruebas de carga, las pruebas de estrés, las pruebas de seguridad y las pruebas de usabilidad, cada una enfocada en verificar aspectos específicos del software y garantizar su calidad.

En cuanto a los otros tipos de pruebas, algunas prácticas incluyen:

Pruebas de carga: Realizar pruebas de carga con cargas de trabajo realistas para verificar el rendimiento del sistema bajo carga.

Pruebas de estrés: Realizar pruebas de estrés para verificar el rendimiento del sistema en situaciones extremas, como picos de tráfico o fallas en componentes críticos.

Pruebas de seguridad: Realizar pruebas de seguridad para verificar que el software sea seguro contra posibles vulnerabilidades.

Pruebas de usabilidad: Realizar pruebas de usabilidad con usuarios finales para verificar que el software sea fácil de usar y cumpla con las expectativas del usuario.

En general, las mejores prácticas para todas las pruebas de software incluyen planificar las pruebas desde el principio del proyecto, documentar los resultados de las pruebas, automatizar las pruebas cuando sea posible, y realizar pruebas regularmente durante todo el ciclo de vida del software.

7.13 Mantenimiento de software

El mantenimiento de software es el proceso de modificar, actualizar y mejorar el software después de su entrega inicial para corregir errores, mejorar el rendimiento, adaptarlo a nuevas necesidades y garantizar su operación continua. En otras palabras, es la actividad que se lleva a cabo después de que se ha entregado el software y que se centra en mantenerlo en funcionamiento de manera efectiva y eficiente.

El mantenimiento de software es una parte importante del ciclo de vida del software y puede ser necesario por varias razones, como cambios en los requisitos del usuario, actualizaciones de hardware o software, problemas de seguridad o errores que necesitan ser corregidos.

Hay tres tipos principales de mantenimiento de software:

Mantenimiento correctivo: se enfoca en corregir errores y fallas en el software.

Mantenimiento preventivo: se enfoca en prevenir problemas futuros y mejorar la calidad del software.

Mantenimiento evolutivo: se enfoca en agregar nuevas funcionalidades y mejorar la capacidad del software para satisfacer las necesidades cambiantes de los usuarios.

En general, el mantenimiento de software es esencial para garantizar que el software siga funcionando de manera efectiva y cumpla con los requisitos del usuario. Para llevar a cabo el mantenimiento de software de manera efectiva, se necesitan técnicas y herramientas específicas, como la evaluación de la calidad del software, el monitoreo del rendimiento y el seguimiento de las incidencias y problemas. Además, es importante realizar un mantenimiento planificado y regular para asegurarse de que el software se mantenga actualizado y funcionando correctamente

El mantenimiento de software es una actividad esencial para garantizar que los sistemas de software sigan funcionando correctamente y satisfagan las necesidades de los usuarios a lo largo del tiempo. El mantenimiento de software se lleva a cabo para corregir errores, mejorar la calidad del software, agregar nuevas funciones y adaptarse a los cambios en los requisitos y tecnologías.

La importancia del mantenimiento de software puede explicarse en los siguientes términos:

Corrección de errores: el mantenimiento de software ayuda a identificar y corregir errores y fallas en el software que pueden afectar su rendimiento y fiabilidad. Esto asegura que el software siga siendo útil y funcional para los usuarios.

Mejora de la calidad: el mantenimiento de software permite a los desarrolladores identificar y corregir problemas de calidad en el software, como problemas de seguridad, diseño deficiente y mala usabilidad. La mejora de la calidad del software puede aumentar la satisfacción del usuario y reducir los costos asociados con la corrección de errores y la resolución de problemas.

Adición de nuevas características: el mantenimiento de software permite agregar nuevas características y funcionalidades al software existente, lo que puede mejorar su valor y utilidad para los usuarios.

Adaptación a cambios: el mantenimiento de software también ayuda a los desarrolladores a adaptarse a los cambios en los requisitos y tecnologías, lo que puede ser necesario para mantener el software actualizado y relevante en un entorno empresarial en constante evolución.

Para llevar a cabo el mantenimiento de software, los desarrolladores deben seguir un proceso sistemático que incluya las siguientes etapas:

Análisis: se lleva a cabo una evaluación del software para identificar problemas y áreas que necesitan mejoras.

Diseño: se desarrollan soluciones para corregir los problemas y mejorar el software.

Implementación: se implementan las soluciones y se prueba el software para asegurarse de que funcione correctamente.

Pruebas: se llevan a cabo pruebas para garantizar que el software siga funcionando correctamente y que los problemas identificados se hayan solucionado adecuadamente.

Mantenimiento: se realiza un seguimiento del software para asegurarse de que siga funcionando correctamente y que cualquier problema que surja sea abordado de inmediato.

En resumen, el mantenimiento de software es una actividad esencial para garantizar que el software siga siendo útil y funcional a lo largo del tiempo. Los desarrolladores deben seguir un proceso sistemático para llevar a cabo el mantenimiento de software, que incluye la identificación y corrección de errores, la mejora de la calidad, la adición de nuevas características y la adaptación a los cambios en los requisitos y tecnologías.

Durante el proceso de mantenimiento de software se realizan las siguientes actividades:

Corrección de errores: Esta actividad se enfoca en identificar y solucionar problemas en el software que impiden que funcione correctamente. La corrección de errores incluye la identificación de los problemas, la implementación de soluciones para resolverlos y la realización de pruebas para verificar que los errores hayan sido corregidos satisfactoriamente. Los errores pueden ser encontrados por los usuarios o pueden ser identificados por los desarrolladores mediante pruebas y análisis de datos.

Adaptación a cambios en los requisitos: Los requisitos del software pueden cambiar con el tiempo debido a factores como los cambios en las necesidades de los usuarios o los cambios en el entorno empresarial. La adaptación a estos cambios implica modificar el software existente para satisfacer las nuevas necesidades y requisitos. Esta actividad puede incluir la adición de nuevas características, la eliminación de características obsoletas o la modificación de características existentes para mejorar su funcionamiento.

Mejora del rendimiento: Esta actividad se enfoca en optimizar el rendimiento del software para mejorar su velocidad, eficiencia y capacidad de manejo de datos. La mejora del rendimiento puede incluir la identificación de cuellos de botella y la implementación de soluciones para mejorar el rendimiento en áreas específicas. Además, puede incluir la optimización del código fuente para reducir el uso de recursos del sistema y mejorar el tiempo de respuesta.

Mantenimiento preventivo: Esta actividad se enfoca en prevenir problemas futuros mediante la implementación de soluciones para evitar la ocurrencia de problemas conocidos. El

mantenimiento preventivo puede incluir la aplicación de parches de seguridad, la actualización de software obsoleto y la limpieza de archivos y registros obsoletos.

En resumen, las diferentes actividades que se realizan durante el proceso de mantenimiento de software incluyen la corrección de errores, la adaptación a cambios en los requisitos, la mejora del rendimiento y el mantenimiento preventivo. Todas estas actividades son importantes para garantizar que el software siga siendo útil y funcional a lo largo del tiempo y satisfaga las necesidades de los usuarios.

Ejemplos de cómo se han llevado a cabo estas actividades en proyectos reales de mantenimiento de software

Corrección de errores: En el proyecto de mantenimiento del sistema de gestión de inventario de una empresa de logística, se identificó un error que causaba que los pedidos no se procesaran correctamente, lo que provocaba retrasos en la entrega de los productos. El equipo de mantenimiento de software analizó el código fuente y encontró el problema. Luego, implementaron una solución que resolvió el problema y realizaron pruebas para asegurarse de que el error ya no existía.

Adaptación a cambios en los requisitos: En un proyecto de mantenimiento de un sistema de gestión de recursos humanos de una empresa de servicios, se identificó que los empleados necesitaban tener acceso al sistema desde dispositivos móviles, ya que cada vez más trabajaban de manera remota. El equipo de mantenimiento de software modificó el sistema existente para que fuera compatible con dispositivos móviles y rediseñaron la interfaz para que fuera más fácil de usar en pantallas más pequeñas.

Mejora del rendimiento: En el proyecto de mantenimiento del sistema de reserva de una aerolínea, se identificó que el sistema estaba experimentando cuellos de botella que causaban retrasos en la respuesta del sistema. El equipo de mantenimiento de software implementó soluciones de optimización de base de datos, reduciendo el tiempo de respuesta del sistema y mejorando la experiencia del usuario.

Mantenimiento preventivo: En el proyecto de mantenimiento de un sistema de punto de venta de una cadena de tiendas minoristas, el equipo de mantenimiento de software aplicó parches de seguridad y actualizó el software obsoleto en el sistema para prevenir futuros problemas de seguridad y para asegurar que el sistema siguiera siendo compatible con las últimas tecnologías.

Estos son solo algunos ejemplos de cómo se llevan a cabo las diferentes actividades de mantenimiento de software en proyectos reales. Cada proyecto es único y puede requerir diferentes actividades de mantenimiento según las necesidades y requisitos del sistema.

7.14 Conclusiones del capítulo

La ingeniería de software es una disciplina que combina teoría y práctica para desarrollar software de alta calidad. Es un proceso sistemático que involucra la planificación, el diseño, la implementación, la prueba y el mantenimiento del software.

Los ingenieros de software utilizan una variedad de herramientas y técnicas para desarrollar software de calidad, incluyendo lenguajes de programación, frameworks, sistemas de control de versiones, bases de datos y herramientas de prueba.

El desarrollo de software es un proceso iterativo en el que los ingenieros de software trabajan en estrecha colaboración con los clientes y los usuarios finales para garantizar que el software cumpla con sus necesidades y requisitos.

La calidad del software es un factor clave en el éxito de cualquier proyecto de software. La ingeniería de software se centra en desarrollar software de calidad que sea fácil de mantener y evolucionar.

La ingeniería de software se aplica en una amplia variedad de campos, incluyendo el desarrollo de aplicaciones móviles, la inteligencia artificial, la robótica y la seguridad informática.

La ingeniería de software es una disciplina esencial en la era digital en la que vivimos. La demanda de software de alta calidad sigue creciendo y los ingenieros de software tienen un papel fundamental en el desarrollo de software que responda a las necesidades y expectativas de los usuarios finales.

CONCLUSIONES GENERALES

En este texto se abordaron temas relevantes y conceptos novedosos sobre los sistemas computacionales. Haciendo referencia a la importancia de extraer el conocimiento de los datos, y utilizar las potencialidades de la Inteligencia Artificial, el Internet de las Cosas, la computación en la nube, el Aprendizaje Automático en función de mejorar todos los procesos actuales de la sociedad, teniendo en cuenta altos estándares de seguridad informática, y ética en la manipulación de los datos. El verdadero valor de los datos radica en la posibilidad de extraer de ellos información útil para la toma de decisiones o la exploración y comprensión de los fenómenos que le dieron lugar. El análisis de datos es importante en ramas como: bioinformática, medicina, economía y finanzas, industria, medio ambiente, entre otras, donde el preprocesamiento de estos es esencial.

Se presentó una visión integral de los algoritmos de aprendizaje automático que se pueden aplicar para mejorar la inteligencia y las capacidades de una aplicación. El aprendizaje automático se ha convertido en una tecnología crucial de ciberseguridad que aprende constantemente mediante el análisis de datos para identificar patrones, detectar mejor el malware en el tráfico cifrado, encontrar amenazas internas, predecir dónde están los vecindarios malos en línea, mantener a las personas seguras mientras navegan o proteger los datos en la nube al descubrir actividades sospechosas.

Los avances recientes en la tecnología de microprocesadores y el software han llevado a una mayor capacidad del hardware básico para ejecutar aplicaciones dentro de máquinas virtuales de manera eficiente. Las máquinas virtuales permiten tanto el aislamiento de aplicaciones del hardware subyacente y otras máquinas virtuales como la personalización de la plataforma para satisfacer las necesidades del usuario final.

REFERENCIAS BIBLIOGRÁFICAS

Abbasi, M., & Rafiee, M. (2020). Efficient parallelisation of the packet classification algorithms on multi-core central processing units using multi-threading application program interfaces. *IET Computers & Digital Techniques*, *14*(6), 313-321. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-cdt.2019.0118

Alguliyev, R., Imamverdiyev, Y., & Sukhostat, L. (2018). Cyber-physical systems and their security issues. *Computers in Industry*, *100*, 212-223. https://www.sciencedirect.com/science/article/pii/S0166361517304244

Altınpulluk, H., Kesim, M., & Kurubacak, G. (2020). The usability of augmented reality in open and distance learning systems: A qualitative Delphi study. *Open Praxis*, *12*(2), 283-307. https://search.informit.org/doi/abs/10.3316/informit.353001659113051

Amaya, M. R. A., Ortega-Jimenez, C. H., Garrido-Vega, P., & Machuca, J. A. (2023). Efecto de la industria 4.0 en cadena de suministro Lean y el rendimiento operativo. *Universidad y Sociedad*, *15*(1), 672-683. https://rus.ucf.edu.cu/index.php/rus/article/view/3584

Ambler, S. W. (2002). Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. John Wiley & Sons.

Ansón Alcolea, A., Ogáyar Sánchez, A., & Spranger Hierro, M. J. (2021). Una herramienta para optimizar el comportamiento de los agentes inteligentes en videojuegos mediante Computación Evolutiva. https://eprints.ucm.es/id/eprint/66911/

Anumula, K., & Raymond, J. (2022). Adware and Spyware Detection Using Classification and Association. Proceedings of International Conference on Deep Learning, Computing and Intelligence: ICDCI 2021,

Asatiani, A., Malo, P., Nagbøl, P. R., Penttinen, E., Rinta-Kahila, T., & Salovaara, A. (2021). Sociotechnical envelopment of artificial intelligence: An approach to organizational deployment of inscrutable artificial intelligence systems. *Journal of the Association for Information Systems (JAIS)*, *22*(2), 325-252. https://pure.itu.dk/da/publications/sociotechnical-envelopment-of-artificial-intelligence-an-approach

Aslan, Ö. A., & Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. *Ieee Access*, *8*, 6249-6271. https://doi.org/10.1109/ACCESS.2019.2963724

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. Computer, 21(5), 61-72.

Boehm, B. W. (1981). Software Engineering Economics. Prentice Hall

Bai, J.-J., Lawall, J., Chen, Q.-L., & Hu, S.-M. (2019). Effective Static Analysis of Concurrency Use-After-Free Bugs in Linux Device Drivers. USENIX Annual Technical Conference,

Baños, R. V., Torrado-Fonseca, M., & Álvarez, M. R. (2019). Análisis de regresión lineal múltiple con SPSS: un ejemplo práctico. *REIRE Revista d'Innovació i Recerca en Educació*, *12*(2), 1–10-11–10. https://revistes.ub.edu/index.php/REIRE/article/view/22704

Barsce, J. C., Martínez, E., & Palombarini, J. (2020). Un Enfoque Jerárquico Bi-capa de Optimización Bayesiana de Hiper-parámetros en Aprendizaje por Refuerzos. *AJEA*(5). https://scholar.archive.org/work/eernjekmv5h3lmnwbs5w7vmgjq/access/wayback/http://rtyc.utn.edu.ar/index.php/ajea/article/download/744/623

Bauwens, J., Ruckebusch, P., Giannoulis, S., Moerman, I., & De Poorter, E. (2020). Over-the-air software updates in the internet of things: An overview of key principles. *IEEE Communications Magazine*, *58*(2), 35-41. https://ieeexplore.ieee.org/abstract/document/8999425/

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development. Agile Alliance.

Bello, S. A., Oyedele, L. O., Akinade, O. O., Bilal, M., Delgado, J. M. D., Akanbi, L. A., Ajayi, A. O., & Owolabi, H. A. (2021). Cloud computing in construction industry: Use cases, benefits and challenges. *Automation in Construction*, *122*, 103441. https://www.sciencedirect.com/science/article/pii/S0926580520310219

Berry, M. W., Mohamed, A., & Yap, B. W. (2019). *Supervised and unsupervised learning for data science*. Springer. https://link.springer.com/content/pdf/10.1007/978-3-030-22475-2.pdf

Blondet, G., Le Duigou, J., & Boudaoud, N. (2019). A knowledge-based system for numerical design of experiments processes in mechanical engineering. *Expert Systems with Applications*, *122*, 289-302. https://www.sciencedirect.com/science/article/pii/S0957417419300120

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. Computer, 21(5), 61-72.

Boehm, B. W. (1981). Software Engineering Economics. Prentice Hall.

Boran, F. E., Akay, D., & Yager, R. R. (2016). An overview of methods for linguistic summarization with fuzzy sets. *Expert Systems with Applications*, *61*, 356-377.

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., & Schoellig, A. P. (2022). Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, *5*, 411-444. https://www.annualreviews.org/doi/abs/10.1146/annurev-control-042920-020211

Cabrera-Llanos, A. I., Ortiz-Arango, F., & Cruz-Aranda, F. (2019). Un modelo de minimización de costos de mantenimiento de equipo médico mediante lógica difusa. *Revista mexicana de economía y finanzas*, *14*(3), 379-396. https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1665-53462019000300379

Cárdenas-Pérez, A., & Echeverría, I. B. (2021). Explicación del crecimiento económico en la Economía Popular y Solidaria mediante la aplicación del modelo econométrico de Regresión Lineal y Múltiple. *Revista Publicando*, *8*(28), 74-84. https://www.revistapublicando.org/revista/index.php/crv/article/view/2163

Cardenas, E. Y. M. (2019). Calidad del servicio de internet y satisfacción del cliente. *Industrial data*, *22*(2), 105-116. https://www.redalyc.org/journal/816/81662532008/81662532008.pdf

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., & Zdeborová, L. (2019). Machine learning and the physical sciences. *Reviews of Modern*

Physics, *91*(4), 045002. https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.91.045002

Caroca, A., Derpich, I., Gatica, G., Alfaro, M., Ruete, D., & Hinojosa, M. (2022). Procedimiento de agrupación de estudiantes según riesgo de abandono para mejorar la gestión estudiantil en educación superior. *Texto Livre: Linguagem e Tecnologia*, *15*, 1-22. https://www.redalyc.org/journal/5771/577170677016/577170677016.pdf

Carrillo, J., Gomis, R., De los Santos, S., Covarrubias, L., & Matus, M. (2020). ¿ Podrán transitar los ingenieros a la Industria 4.0? Análisis industrial en Baja California. *Entreciencias: diálogos en la sociedad del conocimiento*, *8*(22). https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-80642020000100311

Castrillón, O. D., Castillo, L. F., & Castaño, C. E. (2022). Minería de datos aplicada a la detección de cáncer gástrico. *Información tecnológica*, *33*(4), 151-160. https://www.scielo.cl/scielo.php?pid=S0718-07642022000400151&script=sci_arttext&tlng=pt

Cicchetti, A., Ciccozzi, F., & Pierantonio, A. (2019). Multi-view approaches for software and system modelling: a systematic literature review. *Software and Systems Modeling*, *18*, 3207-3233. https://link.springer.com/article/10.1007/s10270-018-00713-w

Claramunt, J. C. (2020). La gestión de la información en el paradigma algorítmico: inteligencia artificial y protección de datos. *MÉI: Métodos de Información*, *11*(21), 42-58. https://dialnet.unirioja.es/descarga/articulo/7966054.pdf

Clifton, J., & Laber, E. (2020). Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, *7*, 279-301. https://www.annualreviews.org/doi/abs/10.1146/annurev-statistics-031219-041220

Cockburn, A. (2007). Agile Software Development: The Cooperative Game. Addison-Wesley Professional.

Contreras Contreras, G. F., Medina Delgado, B., Acevedo Jaimes, B. R., & Guevara Ibarra, D. (2022). Metodología de desarrollo de técnicas de agrupamiento de datos usando aprendizaje automático. *Tecnura*, *26*(72), 5-6. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-921X2022000200005

Chacón-Ramírez, E. A., Cardillo-Albarrán, J. J., & Uribe-Hernández, J. (2020). Industria 4.0 en América Latina: Una ruta para su implantación. *Revista Ingenio*, *17*(1), 28-35. https://revistas.ufps.edu.co/index.php/ingenio/article/view/2386

Chen, L., Chen, P., & Lin, Z. (2020). Artificial intelligence in education: A review. *Ieee Access*, *8*, 75264-75278. https://ieeexplore.ieee.org/abstract/document/9069875/

Chuquilla, A., Guarda, T., & Quiña, G. N. (2019). Ransomware-WannaCry Security is everyone's. 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), Churchman, C. W. (1967). Wicked Problems. Management Science, 14(4), B141-B142.

Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2020). A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, *27*, 1071-1092. https://link.springer.com/article/10.1007/s11831-019-09344-w

Deming, W. E. (1986). Out of the Crisis. MIT Press.

Dian, F. J., Vahidnia, R., & Rahmati, A. (2020). Wearables and the Internet of Things (IoT), applications, opportunities, and challenges: A Survey. *Ieee Access*, *8*, 69200-69211. https://ieeexplore.ieee.org/abstract/document/9058658/

Díaz-Ramírez, J. (2021). Aprendizaje Automático y Aprendizaje Profundo. *Ingeniare. Revista chilena de ingeniería*, *29*(2), 180-181. https://www.scielo.cl/scielo.php?pid=S0718-33052021000200180&script=sci_arttext

Ding, Y., Mishra, N., & Hoffmann, H. (2019). Generative and multi-phase learning for computer systems optimization. Proceedings of the 46th International Symposium on Computer Architecture,

Dogan, A., & Birant, D. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications*, *166*, 114060. https://www.sciencedirect.com/science/article/pii/S095741742030823X

Duque-Méndez, N. D., Ovalle-Carranza, D., & Carrillo-Ramos, Á. (2020). Sistema basado en reglas para la generación personalizada de curso virtual. *TecnoLógicas*, *23*(47), 223-236. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-77992020000100223

Efthymiou, S., Ramos-Calderer, S., Bravo-Prieto, C., Pérez-Salinas, A., García-Martín, D., Garcia-Saez, A., Latorre, J. I., & Carrazza, S. (2021). Qibo: a framework for quantum simulation with hardware acceleration. *Quantum Science and Technology*, *7*(1), 015018. https://iopscience.iop.org/article/10.1088/2058-9565/ac39f5/meta

Elsheikh, A. H., Sharshir, S. W., Abd Elaziz, M., Kabeel, A., Guilan, W., & Haiou, Z. (2019). Modeling of solar energy systems using artificial neural network: A comprehensive review. *Solar Energy*, *180*, 622-639. https://www.sciencedirect.com/science/article/pii/S0038092X19300465

Englander, I., & Wong, W. (2021). *The architecture of computer hardware, systems software, and networking: An information technology approach*. John Wiley & Sons. https://www.google.com/books?hl=es&lr=&id=OkcgEAAAQBAJ&oi=fnd&pg=PR9&dq=operating+systems+concepts&ots=lffLyHxoJM&sig=eJNglEXhEw9xnqecXOpmEpAKANE

Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020). A theoretical analysis of deep Q-learning. Learning for Dynamics and Control,

Fayol, H. (1916). General and Industrial Management. Pitman Publishing.

Franco-Árcega, A., Sobrevilla-Sólis, V. I., de Jesús Gutiérrez-Sánchez, M., García-Islas, L. H., Suárez-Navarrete, A., & Rueda-Soriano, E. (2021). Sistema de enseñanza para la técnica de agrupamiento k-means. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, *9*(Especial), 53-58. https://repository.uaeh.edu.mx/revistas/index.php/icbi/article/view/7384

Fu, X., & Cui, T. J. (2019). Recent progress on metamaterials: From effective medium model to real-time information processing system. *Progress in Quantum Electronics*, *67*, 100223. https://www.sciencedirect.com/science/article/pii/S0079672719300151

Ganesan, A., Alagappan, R., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. H. (2021). Exploiting nil-externality for fast replicated storage. Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles,

Gantt, H. L. (1910). Work, Wages, and Profits: Their Influence on the Cost of Production. The Engineering Magazine Company.

García-García, J., Fernández, N., & Imilpán, I. (2020). Desarrollo del razonamiento probabilístico en profesores de matemáticas mediante simulación computacional. *Revista Paradigma*, *41*(Extra 2), 404-426. http://funes.uniandes.edu.co/23637/

García, G. G., Jiménez, C. R., & Marín, J. A. M. (2020). La trascendencia de la Realidad Aumentada en la motivación estudiantil. Una revisión sistemática y meta-análisis. *Alteridad: revista de educación*, *15*(1), 36-46. https://dialnet.unirioja.es/servlet/articulo?codigo=7390919

García Jiménez, M. E., Becerra Sierra, O. A., & Rivera, J. C. (2021). Un Algoritmo evolutivo híbrido para el problema de programación del taller de flujo permutado con restricciones de turno. *Ingeniare. Revista chilena de ingeniería*, *29*(3), 546-556. https://www.scielo.cl/scielo.php?pid=S0718-33052021000300546&script=sci_arttext

García, R. P., Mariscal, D. C., & Murillo, A. C. (2020). El Papel de BIM en la Industria 4.0. *Cimbra: Revista del Colegio de Ingenieros Técnicos de Obras Públicas*(417), 6-14. https://dialnet.unirioja.es/servlet/articulo?codigo=7527312

Gayathri, R., Usharani, S., Mahdal, M., Vezhavendhan, R., Vincent, R., Rajesh, M., & Elangovan, M. (2023). Detection and Mitigation of IoT-Based Attacks Using SNMP and Moving Target Defense Techniques. *Sensors*, *23*(3), 1708. https://www.mdpi.com/2114040

Giraldo, J., Cardenas, A., & Sanfelice, R. (2023). A switching-based Moving Target Defense against sensor attacks in control systems. *Nonlinear Analysis: Hybrid Systems*, *47*, 101268. https://www.sciencedirect.com/science/article/pii/S1751570X22000681

Gleirscher, M., & Marmsoler, D. (2020). Formal methods in dependable systems engineering: a survey of professionals from Europe and North America. *Empirical Software Engineering*, *25*, 4473-4546. https://link.springer.com/article/10.1007/s10664-020-09836-5

Glielmo, A., Husic, B. E., Rodriguez, A., Clementi, C., Noé, F., & Laio, A. (2021). Unsupervised learning methods for molecular simulation data. *Chemical Reviews*, *121*(16), 9722-9758. https://pubs.acs.org/doi/abs/10.1021/acs.chemrev.0c01195

Gope, P., Gheraibia, Y., Kabir, S., & Sikdar, B. (2020). A secure IoT-based modern healthcare system with fault-tolerant decision making process. *IEEE Journal of Biomedical and Health Informatics*, *25*(3), 862-873. https://ieeexplore.ieee.org/abstract/document/9134756/

Guzmán, R. S. H., De La Rosa, C. G. B., Barrezueta, L. D. R., & Sánchez, P. M. M. (2022). Fundamentos de la auditoría: Una aproximación del estado del arte. *Serie Científica de la Universidad de las Ciencias Informáticas*, *15*(12), 245-266. https://publicaciones.uci.cu/index.php/serie/article/view/1282

Hölldobler, K., Michael, J., Ringert, J. O., Rumpe, B., & Wortmann, A. (2019). Innovations in model-based software and systems engineering. https://www.researchgate.net/profile/Bernhard-Rumpe/publication/334131991_Innovations_in_Model-based_Software_And_Systems_Engineering/links/6267e5dd8e6d637bd1ffdd9d/Innovations-in-Model-based-Software-And-Systems-Engineering.pdf

Huang, H.-Y., Broughton, M., Mohseni, M., Babbush, R., Boixo, S., Neven, H., & McClean, J. R. (2021). Power of data in quantum machine learning. *Nature communications*, *12*(1), 2631. https://www.nature.com/articles/s41467-021-22539-9

Hunt, J., & Hunt, J. (2019). Multiprocessing. *Advanced Guide to Python 3 Programming*, 363-376. https://link.springer.com/chapter/10.1007/978-3-030-25943-3_31

IEEE Standards Association. (2010). IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990 (Revision of IEEE Std 610.12-1990).

Ishikawa, K. (1985). What is Total Quality Control? The Japanese Way. Prentice-Hall.

Jafferis, D., Zlokapa, A., Lykken, J. D., Kolchmeyer, D. K., Davis, S. I., Lauk, N., Neven, H., & Spiropulu, M. (2022). Traversable wormhole dynamics on a quantum processor. *Nature*, *612*(7938), 51-55. https://www.nature.com/articles/s41586-022-05424-3

Jahan, I. S., Snasel, V., & Misak, S. (2020). Intelligent systems for power load forecasting: A study review. *Energies*, *13*(22), 6105. https://www.mdpi.com/897884

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., & Makedon, F. (2020). A survey on contrastive self-supervised learning. *Technologies*, *9*(1), 2. https://www.mdpi.com/2227-7080/9/1/2

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685-695. https://link.springer.com/article/10.1007/s12525-021-00475-2

Javaid, M., & Haleem, A. (2019). Industry 4.0 applications in medical field: A brief review. *Current Medicine Research and Practice*, *9*(3), 102-109. https://www.sciencedirect.com/science/article/pii/S2352081719300418

Junca Peláez, M. J., & Velasco Gregory, M. F. (2020). gradiente estocástico y aproximación estocástica aplicados a Q-learning. https://repositorio.uniandes.edu.co/handle/1992/51295

Juran, J. M. (1988). Quality Control Handbook. McGraw-Hill.

Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligence Review*, *52*(1), 273-292. https://link.springer.com/article/10.1007/s10462-018-09677-1

Keith, A., Ferrada, H., & Navarro, C. A. (2022). Accelerating the Convex Hull Computation with a Parallel GPU Algorithm. 2022 41st International Conference of the Chilean Computer Science Society (SCCC),

Khan, S. A., Naim, I., Kusi-Sarpong, S., Gupta, H., & Idrisi, A. R. (2021). A knowledge-based experts' system for evaluation of digital supply chain readiness. *Knowledge-Based Systems*, *228*, 107262. https://www.sciencedirect.com/science/article/pii/S0950705121005244

Khanna, A., & Kaur, S. (2020). Internet of things (IoT), applications and challenges: a comprehensive review. *Wireless Personal Communications*, *114*, 1687-1762. https://link.springer.com/article/10.1007/s11277-020-07446-4

Kida, T., & Kida, Y. (2022). Theory of the optimum signal approximation clarifying the importance in the recognition of parallel world and application to secure signal communication with feedback. 2022 International Joint Conference on Neural Networks (IJCNN),

Kossiakoff, A., Biemer, S. M., Seymour, S. J., & Flanigan, D. A. (2020). *Systems engineering principles and practice*. John Wiley & Sons. https://www.google.com/books?hl=es&lr=&id=-DXrDwAAQBAJ&oi=fnd&pg=PA15&dq=software+system+engineering&ots=9ibdlF4Z_T&sig=CJgO8EBPXruxG_KGY7QahMQaWzU

Krishnan, R. S., Julie, E. G., Robinson, Y. H., Raja, S., Kumar, R., & Thong, P. H. (2020). Fuzzy logic based smart irrigation system using internet of things. *Journal of Cleaner Production*, *252*, 119902. https://www.sciencedirect.com/science/article/pii/S0959652619347729

Lai, J. W., & Cheong, K. H. (2022). Educational Opportunities and Challenges in Augmented Reality: Featuring Implementations in Physics Education. *IEEE Access*, *10*, 43143-43158. https://ieeexplore.ieee.org/abstract/document/9755165/

Lai, Y.-P., & Hsia, P.-L. (2007). Using the vulnerability information of computer systems to improve the network security. *Computer Communications*, *30*(9), 2032-2047. https://www.sciencedirect.com/science/article/pii/S014036640700117X

Lambora, A., Gupta, K., & Chopra, K. (2019). Genetic algorithm-A literature review. 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon),

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments: A brief history. IEEE Computer, 36(6), 47-56.

Larranaga, P., Lozano, J. A., & Mühlenbein, H. (2003). Algoritmos de estimación de distribuciones en problemas de optimización combinatoria. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, *7*(19), 0. https://www.redalyc.org/pdf/925/92571910.pdf

Larrañaga, P., Lozano, J. A., Mühlenbein, H., & Agustin, S. (2003). Estimation of distribution algorithms applied to combinatorial optimization problems. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, *19*, 149-168. https://www.redalyc.org/pdf/925/92571910.pdf

László, K., & Ghous, H. (2020). Efficiency comparison of Python and RapidMiner. *Multidiszciplináris Tudományok*, *10*(3), 212-220. https://ojs.uni-miskolc.hu/index.php/multi/article/view/539

Leal-Cornejo, F., López-García, R. E., Martínez-Montiel, M. G., Tapia-Castillo, D. I., & de León-Vázquez, I. I. (2019). Análisis de Regresión y Correlación Lineal. *XIKUA Boletín Científico de la Escuela Superior de Tlahuelilpan*, *7*(13), 62-64. https://repository.uaeh.edu.mx/revistas/index.php/xikua/article/view/3558

Leiserson, C. E., Thompson, N. C., Emer, J. S., Kuszmaul, B. C., Lampson, B. W., Sanchez, D., & Schardl, T. B. (2020). There's plenty of room at the Top: What will drive computer performance after Moore's law? *Science*, *368*(6495), eaam9744. https://www.science.org/doi/abs/10.1126/science.aam9744

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. Proceedings of the IEEE, 68(9), 1060-1076.

Lientz, B. P., & Swanson, E. B. (2008). Software maintenance management: Evaluation and continuous improvement. John Wiley & Sons.

Liu, B., Wu, H., Yang, Q., & Zhang, H. (2023). Random-Enabled Hidden Moving Target Defense against False Data Injection Alert Attackers. *Processes*, *11*(2), 348. https://www.mdpi.com/2227-9717/11/2/348

Liu, F., Tang, G., Li, Y., Cai, Z., Zhang, X., & Zhou, T. (2019). A survey on edge computing systems and tools. *Proceedings of the IEEE*, *107*(8), 1537-1562. https://ieeexplore.ieee.org/abstract/document/8746691/

Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., & Shi, W. (2019). Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, *107*(8), 1697-1716. https://ieeexplore.ieee.org/abstract/document/8744265/

Loncaric, F., Cámara, Ó., Piella, G., & Bijnens, B. (2021). La integración de la inteligencia artificial en el abordaje clínico del paciente: Enfoque en la imagen cardiaca. *Revista Española de Cardiología*, *74*(1), 72-80. https://www.sciencedirect.com/science/article/pii/S0300893220304231

Lu, J., Tan, L., & Jiang, H. (2021). Review on convolutional neural network (CNN) applied to plant leaf disease classification. *Agriculture*, *11*(8), 707. https://www.mdpi.com/1204652

Lu, Y. (2019). Artificial intelligence: a survey on evolution, models, applications and future trends. *Journal of Management Analytics*, *6*(1), 1-29. https://www.tandfonline.com/doi/abs/10.1080/23270012.2019.1570365

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, *7*(66), eabm6074. https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

Malallah, H., Zeebaree, S. R., Zebari, R. R., Sadeeq, M. A., Ageed, Z. S., Ibrahim, I. M., Yasin, H. M., & Merceedi, K. J. (2021). A comprehensive study of kernel (issues and concepts) in different operating systems. *Asian Journal of Research in Computer Science*, *8*(3), 16-31. http://openarticledepository.com/id/eprint/128/

Marcillo Sánchez, P. M. (2022). *Análisis del desarrollo de software con metodología ágil y la capacidad de la sostenibilidad implementada* ETSI_Sistemas_Infor]. https://oa.upm.es/id/eprint/71758

Márquez Díaz, J. (2019). Riesgos y vulnerabilidades de la denegación de servicio distribuidos en internet de las cosas. *Revista de Bioética y Derecho*(46), 85-100. https://scielo.isciii.es/scielo.php?pid=S1886-58872019000200006&script=sci_arttext&tlng=pt

Martin, J. (1991). Rapid Application Development. Macmillan Publishing.

Martínez-Plumed, F., Contreras-Ochando, L., Ferri, C., Hernández-Orallo, J., Kull, M., Lachiche, N., Ramirez-Quintana, M. J., & Flach, P. (2019). CRISP-DM twenty years later: From data mining processes to data science trajectories. *IEEE Transactions on Knowledge and Data Engineering*, *33*(8), 3048-3061. https://ieeexplore.ieee.org/abstract/document/8943998/

Martínez, A. E. C., & Huertas, C. E. B. (2021). Caracterización de la tendencia del COVID-19 en Colombia con regresiones polinomiales. *Gerencia y Políticas de Salud*, *20*, 1-12. https://revistas.javeriana.edu.co/index.php/gerepolsal/article/view/32111/26780

Méndez-Gurrola, I. I. (2020). Aprendizaje automático aplicado en física: Una revisión de la literatura científica. *Instituto de Arquitectura Diseño y Arte*. http://cathi.uacj.mx/handle/20.500.11961/18429

Mendonça, J., Cho, J.-H., Moore, T. J., Nelson, F. F., Lim, H., & Dongseong Kim, D. (2023). Performance impact analysis of services under a time-based moving target defense mechanism. *The Journal of Defense Modeling and Simulation*, *20*(1), 41-56. https://journals.sagepub.com/doi/abs/10.1177/15485129211036937

Merlini, D., & Rossini, M. (2021). Text categorization with WEKA: A survey. *Machine Learning with Applications*, *4*, 100033. https://www.sciencedirect.com/science/article/pii/S2666827021000141

Miscuglio, M., & Sorger, V. J. (2020). Photonic tensor cores for machine learning. *Applied Physics Reviews*, *7*(3), 031404. https://aip.scitation.org/doi/abs/10.1063/5.0001942

Mishra, R., Barnwal, S. K., Malviya, S., Singh, V., Singh, P., Singh, S., & Tiwary, U. S. (2020). Computing with words through interval type-2 fuzzy sets for decision making environment. Intelligent Human Computer Interaction: 11th International Conference, IHCI 2019, Allahabad, India, December 12–14, 2019, Proceedings 11,

Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, *16*(1), 1-118. https://www.nowpublishers.com/article/Details/MAL-086

Morales España, G., Mora Flórez, J., & Vargas Torres, H. (2008). Estrategia de regresión basada en el método de los k vecinos más cercanos para la estimación de la distancia de falla en sistemas radiales. *Revista Facultad de Ingeniería Universidad de Antioquia*(45), 100-108. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302008000300009

Mulloni, V., & Donelli, M. (2020). Chipless RFID sensors for the Internet of Things: Challenges and opportunities. *Sensors*, *20*(7), 2135. https://www.mdpi.com/687556

Musgrave, J., Purdy, C., Ralescu, A. L., Kapp, D., & Kebede, T. (2020). Semantic feature discovery of trojan malware using vector space kernels. 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS),

Musleh, M. M., Alajrami, E., Khalil, A. J., Abu-Nasser, B. S., Barhoom, A. M., & Naser, S. A. (2019). Predicting liver patients using artificial neural network. *International Journal of Academic Information Systems Research (IJAISR)*, *3*(10). https://core.ac.uk/download/pdf/266990162.pdf

Naeem, M., Chaudhry, S. A., Mahmood, K., Karuppiah, M., & Kumari, S. (2020). A scalable and secure RFID mutual authentication protocol using ECC for Internet of Things. *International Journal of Communication Systems*, *33*(13), e3906. https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.3906

Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N. R., Ganger, G. R., Gibbons, P. B., & Zaharia, M. (2019). PipeDream: Generalized pipeline parallelism for DNN training. Proceedings of the 27th ACM Symposium on Operating Systems Principles,

Nauman, A., Qadri, Y. A., Amjad, M., Zikria, Y. B., Afzal, M. K., & Kim, S. W. (2020). Multimedia Internet of Things: A comprehensive survey. *Ieee Access*, *8*, 8202-8250. https://ieeexplore.ieee.org/abstract/document/8950450/

Neese, F. (2022). Software update: The ORCA program system—Version 5.0. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *12*(5), e1606. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1606

Ngiam, K. Y., & Khor, W. (2019). Big data and machine learning algorithms for health-care delivery. *The Lancet Oncology*, *20*(5), e262-e273. https://www.sciencedirect.com/science/article/pii/S1470204519301494

Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., Niyato, D., Dobre, O., & Poor, H. V. (2021). 6G Internet of Things: A comprehensive survey. *IEEE Internet of Things Journal*, *9*(1), 359-383. https://ieeexplore.ieee.org/abstract/document/9509294/

Nguyen, T., Gosine, R. G., & Warrian, P. (2020). A systematic review of big data analytics for oil and gas industry 4.0. *Ieee Access*, *8*, 61183-61201. https://ieeexplore.ieee.org/abstract/document/9028154/

Ni, K., Yin, X., Laguna, A. F., Joshi, S., Dünkel, S., Trentzsch, M., Müller, J., Beyer, S., Niemier, M., & Hu, X. S. (2019). Ferroelectric ternary content-addressable memory for one-shot learning. *Nature Electronics*, *2*(11), 521-529. https://www.nature.com/articles/s41928-019-0321-3

Ni, L., Wang, D., Wu, J., Wang, Y., Tao, Y., Zhang, J., & Liu, J. (2020). Streamflow forecasting using extreme gradient boosting model coupled with Gaussian mixture model. *Journal of Hydrology*, *586*, 124901. https://www.sciencedirect.com/science/article/pii/S0022169420303619

Niño, F. Y. A. (2023). Ransomware, una amenaza latente en Latinoamérica. *InterSedes*, *24*(49), 92-119. https://revistas.ucr.ac.cr/index.php/intersedes/article/view/50765

Noori, D., Shakeri, H., & Niazi Torshiz, M. (2020). Scalable, efficient, and secure RFID with elliptic curve cryptosystem for Internet of Things in healthcare environment. *EURASIP Journal on Information Security*, *2020*, 1-11. https://link.springer.com/article/10.1186/s13635-020-00114-x

Norouzi, M., Heikalabad, S. R., & Salimzadeh, F. (2020). A reversible ALU using HNG and Ferdkin gates in QCA nanotechnology. *International Journal of Circuit Theory and Applications*, *48*(8), 1291-1303. https://onlinelibrary.wiley.com/doi/abs/10.1002/cta.2799

Ntoutsi, E., Fafalios, P., Gadiraju, U., Iosifidis, V., Nejdl, W., Vidal, M. E., Ruggieri, S., Turini, F., Papadopoulos, S., & Krasanakis, E. (2020). Bias in data-driven artificial intelligence systems—An introductory survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *10*(3), e1356. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1356

Ospina-Gutiérrez, J. P., & Aristizábal, E. (2021). Aplicación de inteligencia artificial y técnicas de aprendizaje automático para la evaluación de la susceptibilidad por movimientos en masa. *Revista Mexicana De Ciencias Geológicas*, *38*(1), 43-54. https://geohazards.com.co/documentos/publicacion1.pdf

Ospina Díaz, M. R., & Sanabria Rangel, P. E. (2020). Desafíos nacionales frente a la ciberseguridad en el escenario global: un análisis para Colombia. *Revista Criminalidad*,

62(2), 199-217. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-31082020000200199

Ouali, Y., Hudelot, C., & Tami, M. (2020). An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*. https://arxiv.org/abs/2006.05278

Paccagnella, R., Datta, P., Hassan, W. U., Bates, A., Fletcher, C., Miller, A., & Tian, D. (2020). Custos: Practical tamper-evident auditing of operating systems using trusted execution. Network and distributed system security symposium,

Pan, C., Ren, H., Wang, K., Kolb, J. F., Elkashlan, M., Chen, M., Di Renzo, M., Hao, Y., Wang, J., & Swindlehurst, A. L. (2021). Reconfigurable intelligent surfaces for 6G systems: Principles, applications, and research directions. *IEEE Communications Magazine*, *59*(6), 14-20. https://ieeexplore.ieee.org/abstract/document/9475160/

Pardo, M. L.-P., Carrera, F. R., Cerqueiras, V. B., Romero, E. R., Dalí, A. P., Abellón, M. R., Linares, S. L., & Calvo, B. R. (2022). Sistema inteligente para la gestión de la demanda en atención primaria. *Journal of Healthcare Quality Research*. https://www.sciencedirect.com/science/article/pii/S260364792200094X

Partel, V., Kakarla, S. C., & Ampatzidis, Y. (2019). Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence. *Computers and electronics in agriculture*, *157*, 339-350. https://www.sciencedirect.com/science/article/pii/S0168169918316612

Pérez Esteras, J. (2020). Uso de técnicas de minería de datos para la detección de anomalías y predicción de la producción en una turbina eólica. https://openaccess.uoc.edu/handle/10609/118007

Pérez Rave, J. I. (2019). Statihouse®: desarrollo tecnológico basado en ciencia de datos para explorar estadísticamente el sector inmobiliario. *Ingeniare. Revista chilena de ingeniería*, *27*(1), 113-130. https://www.scielo.cl/scielo.php?pid=S0718-33052019000100113&script=sci_arttext&tlng=pt

Pham, H. V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., Yu, Y., & Nagappan, N. (2020). Problems and opportunities in training deep learning software systems: An analysis of variance. Proceedings of the 35th IEEE/ACM international conference on automated software engineering,

Piñero, J., Ramírez-Anguita, J. M., Saüch-Pitarch, J., Ronzano, F., Centeno, E., Sanz, F., & Furlong, L. I. (2020). The DisGeNET knowledge platform for disease genomics: 2019 update. *Nucleic acids research*, *48*(D1), D845-D855. https://academic.oup.com/nar/article-abstract/48/D1/D845/5611674

Pressman, R. S., & Maxim, B. R. (2015). Software engineering: A practitioner's approach. McGraw Hill Education. (Capítulo 21: Software maintenance)

Pressman, R. S. (2014). Ingeniería del software: un enfoque práctico (7ma edición). McGraw Hill.

Putro, M. D., Kurnianggoro, L., & Jo, K.-H. (2020). High performance and efficient real-time face detector on central processing unit based on convolutional neural network. *IEEE Transactions on Industrial Informatics*, *17*(7), 4449-4457. https://ieeexplore.ieee.org/abstract/document/9187678/

Qi, Q., & Tao, F. (2019). A smart manufacturing service system based on edge computing, fog computing, and cloud computing. *Ieee Access*, *7*, 86769-86777. https://ieeexplore.ieee.org/abstract/document/8740963/

Quezada-Sarmiento, P. A., & Suárez Guerrero, C. (2021). La Computación en la Nube en el proceso formativo en Programación Web. *RISTI: Revista Ibérica de Sistemas e Tecnologias de Informação, 2021, num. E42, p. 10-19*. https://roderic.uv.es/handle/10550/82609

Rahman, A. U., Saeed, M., Khan, K. A., & Matendo Mabela, R. (2022). Set-theoretic inequalities based on convex multi-argument approximate functions via set inclusion. *Journal of Function Spaces*, *2022*. https://www.hindawi.com/journals/jfs/2022/6998104/

Ramírez, R. O., Castillo, M. P., Arjonilla, J. E., Suárez, O. Y., Zerón, H. M., & Lagos, J. R. (2021). Bosque Aleatorio Basado en Rasgos Tiempo-Frecuencia de la VFC Fetal para la Identificación de Actividad Uterina en Fetos a Término y Pretérmino. Memorias del Congreso Nacional de Ingeniería Biomédica,

[Record #423 is using a reference type undefined in this output style.]

Rathee, G., Balasaraswathi, M., Chandran, K. P., Gupta, S. D., & Boopathi, C. (2021). A secure IoT sensors communication in industry 4.0 using blockchain technology. *Journal of Ambient Intelligence and Humanized Computing*, *12*, 533-545. https://link.springer.com/article/10.1007/s12652-020-02017-8

Ravindran, K., Iannelli, M., & Adiththan, A. (2023). Moving-Target-Defense based Security Mechanisms: A System Management Perspective. 2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS),

Rendón-Hurtado, N.-D., Isaza-Narváez Ph, C.-V., & Rodríguez-Buriticá Ph, S. (2020). Identificación automática de transformación en el bosque seco tropical colombiano usando GMM y UBM-GMM. *Revista Facultad de Ingeniería*, *29*(54). http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-11292020000100032

Ricardo, J. E., Vázquez, M. Y. L., Palacios, A. J. P., & Ojeda, Y. E. A. (2021). Inteligencia artificial y propiedad intelectual. *Universidad y Sociedad*, *13*(S3), 362-368. https://rus.ucf.edu.cu/index.php/rus/article/view/2490

Rodríguez, A., Castro, V. F. R., Gonzalez, A. D. C. R., Baque, N. A. C., & Tarragó, J. C. P. (2021). Aplicaciones de la Inteligencia Artificial en técnicas de minería de procesos. *Serie Científica de la Universidad de las Ciencias Informáticas*, *14*(7), 136-155. https://dialnet.unirioja.es/servlet/articulo?codigo=8590663

Rodriguez, J. A., & Celis, O. (2019). Sistema computacional de análisis de la vulnerabilidad en puentes de la red vial nacional en Colombia utilizando el modelo de jerarquización analítica. *Ingenio Magno*, *10*(1), 111-130. https://dialnet.unirioja.es/servlet/articulo?codigo=7537076

Rodriguez, M. A., Romero, M. F. L., Henríquez, L. M. T., & Buitrago, G. A. F. (2021). Modelo de aprendizaje automático como herramienta para la toma de decisiones en la cuenca del río Ariporo. *INVENTUM*, *16*(31), 15-23. https://revistas.uniminuto.edu/index.php/Inventum/article/view/2765

Rojas, E. M. (2020). Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo. *Revista Ibérica de Sistemas e Tecnologias de Informação*(E28), 586-

599. https://search.proquest.com/openview/c7e24c997199215aa26a39107dd2fe98/1?pq-origsite=gscholar&cbl=1006393

Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques. Proceedings of IEEE WESCON, 1-9.

Ruan, J., & Li, H. (2020). Fast and accurate long-read assembly with wtdbg2. *Nature methods*, *17*(2), 155-158. https://www.nature.com/articles/s41592-019-0669-3

Ruelas, E., López, J. A. V., Salgado, J. C., Márquez, J. A. S., Serrato, R. B., & García, J. A. J. (2020). Control estadistico de procesos multivariantes a través de la red neuronal artificial perceptron multicapa y el análisis del grafico mewma. *IEEE Latin America Transactions*, *18*(6), 1041-1048. http://latamt.ieeer9.org/index.php/transactions/article/view/1284

Ruiz Montezuma, N. D., & Ospina Cadena, W. A. (2021). Algoritmo genético como método de solución al despacho hidrotérmico de corto plazo. https://ciencia.lasalle.edu.co/ing_electrica/619/

Ruiz Rivera, M. E., & Ruiz Lizama, E. (2021). Método de búsqueda eficiente para resolver el problema de identificación de huella dactilar aplicando machine learning. *Industrial data*, *24*(2), 293-317. http://www.scielo.org.pe/scielo.php?script=sci_arttext&pid=S1810-99932021000200293

Sadeeq, M. M., Abdulkareem, N. M., Zeebaree, S. R., Ahmed, D. M., Sami, A. S., & Zebari, R. R. (2021). IoT and Cloud computing issues, challenges and opportunities: A review. *Qubahan Academic Journal*, *1*(2), 1-7. https://journal.qubahan.com/index.php/qaj/article/view/36

Sánchez, P. M. M., & Barrezueta, L. D. R. (2023). Centros de datos verdes en Ecuador: Una estrategia para disminuir la emisión de CO2 en los Centros de Datos ecuatorianos. *Serie Científica de la Universidad de las Ciencias Informáticas*, *16*(1), 1-18. https://publicaciones.uci.cu/index.php/serie/article/view/1229

Sandoya, F. (2022). La analítica y la ciencia de datos en la formación profesional en Ecuador. *Revista Ecuatoriana de Investigación Educativa*, *1*(1), 7-18. https://journal.espe.edu.ec/ojs/index.php/investigacion-educativa/article/view/1506

Schwaber, K. (2004). Agile Project Management with Scrum. Microsoft Press.

Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R., & Eleftheriou, E. (2020). Memory devices and applications for in-memory computing. *Nature nanotechnology*, *15*(7), 529-544. https://www.nature.com/articles/s41565-020-0655-z

Seritan, S., Bannwarth, C., Fales, B. S., Hohenstein, E. G., Isborn, C. M., Kokkila-Schumacher, S. I., Li, X., Liu, F., Luehr, N., & Snyder Jr, J. W. (2021). TeraChem: A graphical processing unit-accelerated electronic structure package for large-scale ab initio molecular dynamics. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *11*(2), e1494. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1494

Shafri, H. Z. M. S. M., & París, C. A. (2019). Métodos de inteligencia artificial (IA) para aplicaciones de teledetección de palma de aceite. *Palmas*, *40*(Especial T), 185-193. https://publicaciones.fedepalma.org/index.php/palmas/article/view/13048

Singh, R. P., Javaid, M., Haleem, A., & Suman, R. (2020). Internet of things (IoT) applications to fight against COVID-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, *14*(4), 521-524. https://www.sciencedirect.com/science/article/pii/S1871402120301065

Sittón-Candanedo, I., Alonso, R. S., Corchado, J. M., Rodríguez-González, S., & Casado-Vara, R. (2019). A review of edge computing reference architectures and a new global edge proposal. *Future Generation Computer Systems*, *99*, 278-294. https://www.sciencedirect.com/science/article/pii/S0167739X1930264X

Smith, S., & Smith, S. (2022). Multiprocessing. *RP2040 Assembly Language Programming: ARM Cortex-M0+ on the Raspberry Pi Pico*, 241-263. https://link.springer.com/chapter/10.1007/978-1-4842-7753-9_13

Smits, G., Nerzic, P., Lesot, M., & Pivert, O. (2019, 23-26 June 2019). FRELS: Fast and Reliable Estimated Linguistic Summaries. 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),

Smits, G., Yager, R. R., & Pivert, O. (2017, 9-12 July 2017). Interactive data exploration on top of linguistic summaries. 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE),

Sommerville, I. (2011). Software engineering. Pearson Education Limited. (Capítulo 27: Maintenance)

Sommerville, I. (2016). Ingeniería de software (10ma edición). Pearson.

Sommerville, I. (2011). Software engineering. Pearson Education Limited. (Capítulo 27: Maintenance)

Sriram, G. (2022). Edge computing vs. Cloud computing: an overview of big data challenges and opportunities for large enterprises. *International Research Journal of Modernization in Engineering Technology and Science*, *4*(1), 1331-1337. https://www.irjmets.com/uploadedfiles/paper/issue_1_january_2022/18590/final/fin_irjmets1643224039.pdf

Suárez, J. N. (2020). Persistencia y regionalización del desplazamiento forzado en Colombia (2005-2010; 2011-2016): una aproximación desde el análisis espacial. *Revista CIFE: lecturas de economía social*, *22*(36), 69-97. https://revistas.usantotomas.edu.co/index.php/cife/article/view/5438

Sunyaev, A., & Sunyaev, A. (2020). Cloud computing. *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, 195-236. https://link.springer.com/chapter/10.1007/978-3-030-34957-8_7

Sutherland, J., & Schwaber, K. (2011). The Scrum guide. Scrum.org.

Sutherland, J. (2014). Scrum: The Art of Doing Twice the Work in Half the Time. Crown Business.

Tamir, D., Neumann, S., Rishe, N., Kandel, A., & Zadeh, L. (2019). Computing with Words—A Framework for Human-Computer Interaction. Augmented Cognition: 13th International Conference, AC 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings 21,

Taylor, F. W. (1911). Principles of Scientific Management. Harper & Brothers.

Tetzlaff, T., Gaudiani, A. A., Rojas Paredes, A., Encinas, D., Fassio, E., Trigila, M., González, R., & Bertaccini, D. (2021). Metaheurísticas, búsqueda estocástica y cómputo eficiente en optimización aplicada. XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja),

Thakkar, A., & Lohiya, R. (2020). A review of the advancement in intrusion detection datasets. *Procedia computer science*, *167*, 636-645. https://www.sciencedirect.com/science/article/pii/S1877050920307961

Tigga, N. P., & Garg, S. (2020). Prediction of type 2 diabetes using machine learning classification methods. *Procedia computer science*, *167*, 706-716. https://www.sciencedirect.com/science/article/pii/S1877050920308024

Timarán-Pereira, R., Caicedo-Zambrano, J., & Hidalgo-Troya, A. (2019). Árboles de decisión para predecir factores asociados al desempeño académico de estudiantes de bachillerato en las pruebas Saber 11. *Revista de investigación, desarrollo e innovación*, *9*(2), 363-378. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S2027-83062019000100363

Torres, J. I. S., & Cardenas, E. G. (2021). Análisis y aplicación de algoritmos de minería de datos. *Revista Perspectivas*, *6*(21), 71-88. https://revistas.uniminuto.edu/index.php/Pers/article/view/2547

Torres, M. M. E., & Manjarrés-Betancur, R. (2020). Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural. *Revista Politécnica*, *16*(31), 85-96. https://www.redalyc.org/journal/6078/607863449007/607863449007.pdf

Trist, E. L. (1981). The Evolution of Socio-Technical Systems: A Conceptual Framework and an Action Research Program. Occasional Paper 2. University of Pennsylvania, Wharton School.

Urbina, R. O. E., Barsallo, E. C., Flores, F. A. I., Villarroel, R. Á. F., Paiva, J. O. N., & Alva, E. E. R. (2021). Método de Montecarlo como estrategia didáctica intercultural para la enseñanza universitaria de la física y matemática en el contexto de la educación no presencial. *Apuntes Universitarios*, *11*(4), 250-268. https://apuntesuniversitarios.upeu.edu.pe/index.php/revapuntes/article/view/770

Valencia, A., & Portilla, P. (2019). Internet Industrial de las Cosas (IIOT): Nueva Forma de Fabricación Inteligente. *Grade Thesis, Fundación Universitaria de Popayán, Colombia*. http://unividafup.edu.co/repositorio/files/original/0cba2296f09e033fe6c5c08e5a6a0119.pdf

Van Santen, J. A., Jacob, G., Singh, A. L., Aniebok, V., Balunas, M. J., Bunsko, D., Neto, F. C., Castaño-Espriu, L., Chang, C., & Clark, T. N. (2019). The natural products atlas: an open access knowledge base for microbial natural products discovery. *ACS central science*, *5*(11), 1824-1833. https://pubs.acs.org/doi/abs/10.1021/acscentsci.9b00806

Vásquez-Quispesivana, W., Inga, M., & Betalleluz-Pallardel, I. (2022). Inteligencia artificial en acuicultura: fundamentos, aplicaciones y perspectivas futuras. *Scientia Agropecuaria*, *13*(1), 79-96. http://www.scielo.org.pe/scielo.php?pid=S2077-99172022000100079&script=sci_arttext

Velásquez, J. D., Olaya, Y., & Franco, C. J. (2010). Predicción de series temporales usando máquinas de vectores de soporte. *Ingeniare. Revista chilena de ingeniería*, *18*(1), 64-75. https://www.scielo.cl/scielo.php?pid=S0718-33052010000100008&script=sci_arttext

Vera, J. C. D., Ortiz, G. M. N., Molina, C., & Vila, M. A. (2019). Extending knowledge based redundancy in association rules with imprecise knowledge. *IEEE Latin America Transactions*, *17*(04), 648-653. https://ieeexplore.ieee.org/abstract/document/8891930/

Verma, D., Singh, K. R., Yadav, A. K., Nayak, V., Singh, J., Solanki, P. R., & Singh, R. P. (2022). Internet of things (IoT) in nano-integrated wearable biosensor devices for healthcare applications. *Biosensors and Bioelectronics: X*, *11*, 100153. https://www.sciencedirect.com/science/article/pii/S2590137022000486

Villamil, S., Hernández, C., & Tarazona, G. (2020). An overview of internet of things. *Telkomnika (Telecommunication Computing Electronics and Control)*, *18*(5), 2320-2327. http://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/15911

Vite Cevallos, H., Carvajal Romero, H., & Barrezueta Unda, S. (2020). Aplicación de algoritmos de aprendizaje automático para clasificar la fertilidad de un suelo bananero. *Conrado*, *16*(72), 15-19. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1990-86442020000100015

Wan, C., Zheng, H., Guo, L., Xu, X., Zhong, R. Y., & Yan, F. (2020). Cloud manufacturing in China: A review. *International Journal of Computer Integrated Manufacturing*, *33*(3), 229-251. https://www.tandfonline.com/doi/abs/10.1080/0951192X.2020.1718768

Wendler, T., & Gröttrup, S. (2021). Using R with the Modeler. In *Data Mining with SPSS Modeler: Theory, Exercises and Solutions* (pp. 1089-1146). Springer. https://link.springer.com/chapter/10.1007/978-3-030-54338-9_9

Wohlgenannt, I., Simons, A., & Stieglitz, S. (2020). Virtual reality. *Business & Information Systems Engineering*, *62*, 455-461. https://link.springer.com/article/10.1007/s12599-020-00658-9

Wu, J., Xiao, P., Huang, H., Gou, F., Zhou, Z., & Dai, Z. (2022). An artificial intelligence multiprocessing scheme for the diagnosis of osteosarcoma MRI images. *IEEE Journal of Biomedical and Health Informatics*, *26*(9), 4656-4667. https://ieeexplore.ieee.org/abstract/document/9802666/

Xiao, Y., Jia, Y., Liu, C., Cheng, X., Yu, J., & Lv, W. (2019). Edge computing security: State of the art and challenges. *Proceedings of the IEEE*, *107*(8), 1608-1631. https://ieeexplore.ieee.org/abstract/document/8741060/

Xie, Z., Bailey, A., Kuleshov, M. V., Clarke, D. J., Evangelista, J. E., Jenkins, S. L., Lachmann, A., Wojciechowicz, M. L., Kropiwnicki, E., & Jagodnik, K. M. (2021). Gene set knowledge discovery with Enrichr. *Current protocols*, *1*(3), e90. https://currentprotocols.onlinelibrary.wiley.com/doi/abs/10.1002/cpz1.90

Xiong, J., Hsiang, E.-L., He, Z., Zhan, T., & Wu, S.-T. (2021). Augmented reality and virtual reality displays: emerging technologies and future perspectives. *Light: Science & Applications*, *10*(1), 216. https://www.nature.com/articles/s41377-021-00658-8

Yang, C. H., Leon, R., Hwang, J., Saraiva, A., Tanttu, T., Huang, W., Camirand Lemyre, J., Chan, K. W., Tan, K., & Hudson, F. E. (2020). Operation of a silicon quantum processor

unit cell above one kelvin. *Nature*, *580*(7803), 350-354. https://www.nature.com/articles/s41586-020-2171-6

Yoo, H., Kim, B., Kim, J. W., & Lee, J. H. (2021). Reinforcement learning based optimal control of batch processes using Monte-Carlo deep deterministic policy gradient with phase segmentation. *Computers & Chemical Engineering*, *144*, 107133. https://www.sciencedirect.com/science/article/pii/S0098135420307912

Yourdon, E. (1993). Decline and Fall of the American Programmer. Prentice-Hall.

Zainab, A., Syed, D., Ghrayeb, A., Abu-Rub, H., Refaat, S. S., Houchati, M., Bouhali, O., & Lopez, S. B. (2021). A multiprocessing-based sensitivity analysis of machine learning algorithms for load forecasting of electric power distribution system. *Ieee Access*, *9*, 31684-31694. https://ieeexplore.ieee.org/abstract/document/9354830/

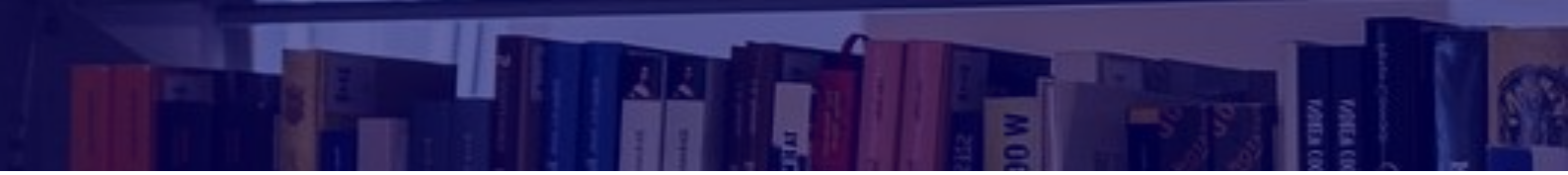
Zhan, T., Yin, K., Xiong, J., He, Z., & Wu, S.-T. (2020). Augmented reality and virtual reality displays: perspectives and challenges. *Iscience*, *23*(8), 101397. https://www.sciencedirect.com/science/article/pii/S258900422030585X

Zhang, F., Parayath, N., Ene, C., Stephan, S., Koehne, A., Coon, M., Holland, E., & Stephan, M. (2019). Genetic programming of macrophages to perform anti-tumor functions using targeted mRNA nanocarriers. *Nature communications*, *10*(1), 3974. https://www.nature.com/articles/s41467-019-11911-5

Zhang, Y., Lan, X., Ren, J., & Cai, L. (2020). Efficient computing resource sharing for mobile edge-cloud computing networks. *IEEE/ACM Transactions on Networking*, *28*(3), 1227-1240. https://ieeexplore.ieee.org/abstract/document/9046758/

Zhao, Y., Chen, X., & Yin, J. (2019). Adaptive boosting-based computational model for predicting potential miRNA-disease associations. *Bioinformatics*, *35*(22), 4730-4738. https://academic.oup.com/bioinformatics/article-abstract/35/22/4730/5481952

Zheng, T., Chen, G., Wang, X., Chen, C., Wang, X., & Luo, S. (2019). Real-time intelligent big data processing: technology, platform, and applications. *Science China Information Sciences*, *62*, 1-12. https://link.springer.com/article/10.1007/s11432-018-9834-8



ISBN: 978-9942-7134-7-6

EDITORIAL ALEMA

Casa Editora