

Ciencia de Datos

Teoría y Aplicaciones

```
{"coord":{"lon":-101.1844,"lat":19.7008},"weather":[{"id":801,"main":"Clouds","description":"fewclouds","icon":"02n"}],  
"base":"stations","main":{"temp":27,"feels_like":26.27,"temp_min":27,"temp_max":27,"pressure":1018,"humidity":26},  
"visibility":6437,"wind":{"speed":6.17,"deg":180},"clouds":{"all":20},"dt":1619659340,"sys":{"type":1,"id":7139,  
"country":"MX","sunrise":1619612246,"sunset":1619658400},"timezone":-18000,"id":3995402,"name":"Morelia","cod":200}  
{"coord":{"lon":-101.1844,"lat":19.7008},"weather":[{"id":801,"main":"Clouds","description":"fewclouds","icon":"02n"}]  
],"base":"stations","main":{"temp":27,"feels_like":26.27,"temp_min":27,"temp_max":27,"pressure":1018,"humidity":26  
},"visibility":6437,"wind":{"speed":6.17,"deg":180},"clouds":{"all":20},"dt":1619659340,"sys":{"type":1,"id":7139,  
"country":"MX","sunrise":1619612246,"sunset":1619658400},"timezone":-18000,"id":3995402,"name":"Morelia","cod":200}  
{"coord":{"lon":-101.1844,"lat":19.7008},"weather":[{"id":801,"main":"Clouds","description":"fewclouds","icon":"02n"}]  
],"base":"stations","main":{"temp":27,"feels_like":26.27,"temp_min":27,"temp_max":27,"pressure":1018,"humidity":26  
},"visibility":6437,"wind":{"speed":6.17,"deg":180},"clouds":{"all":20},"dt":1619659340,"sys":{"type":1,"id":7139,  
"country":"MX","sunrise":1619612246,"sunset":1619658400},"timezone":-18000,"id":3995402,"name":"Morelia","cod":200}
```

Rogelio Ferreira Escutia



TECNOLÓGICO
NACIONAL DE MÉXICO®



CARTA DE RECONOCIMIENTO DEL AUTOR DE LOS DERECHOS A FAVOR DEL TECNM

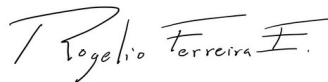
Ciudad de México, 31 / agosto / 2022

TECNOLÓGICO NACIONAL DE MÉXICO. PRESENTE

Bajo protesta de decir verdad, **ROGELIO FERREIRA ESCUTIA**, personal docente adscrito al **CAMPUS MORELIA** del Tecnológico Nacional de México, manifiesto que en cumplimiento de mis actividades relacionadas con el Año Sabático elaboré la obra titulada “**CIENCIA DE DATOS, TEORÍA Y APLICACIONES**”.

Con base en lo anterior, y con fundamento en los artículos 83 de la Ley Federal del Derecho de Autor y 46 de su Reglamento, reconozco que el Tecnológico Nacional de México es titular de los derechos patrimoniales sobre la misma y le corresponden las facultades relativas a la divulgación, integridad de la obra y de colección, conservando el derecho a figurar como autor. Asimismo, respondo por la autoría y originalidad de la citada obra; y relevo de toda responsabilidad al Tecnológico Nacional de México de cualquier demanda o reclamación que llegara a formular alguna persona física o moral que considere que con esta obra es afectado en alguno de los derechos protegidos por la Ley en cita, asumiendo todas las consecuencias legales y económicas.

ATENTAMENTE



Rogelio Ferreira Escutia

Contribución

La “*Ciencia de Datos*” es una nueva disciplina de gran auge en nuestros días, debido al crecimiento exponencial de la información que se genera por medio de los dispositivos y sistemas digitales que manejamos todos los días. Tanto usuarios finales, como empresas de todos tamaños y gobiernos, generan diariamente una gran cantidad de información, que actualmente no se ha explotado adecuadamente todo su potencial. Esto es debido a que, por ser una nueva disciplina, se conoce muy poco y por lo tanto hay pocos profesionistas que tengan este perfil para poder manejar la cantidad de datos que se generan todos los días.

La “*Ciencia de Datos*” está entrando prácticamente en todas las áreas y ámbitos del conocimiento humano, y el poder entender esta información, nos permitirá hacer grandes avances en tecnología, así como en la toma de mejores decisiones en áreas críticas como pueden ser el medio ambiente, la alimentación, la salud y la seguridad. De ahí radica su importancia de trabajar en esta área.

Este trabajo será de gran ayuda y apoyo para orientar nuevos planes de estudio, especialmente de nuevas especialidades que llevarán nuestros alumnos de las carreras de “*Ingeniería en Sistemas Computacionales*” y la “*Ingeniería en Tecnologías de la Información y Comunicaciones*” (aunque podrían aplicarse a cualquier otra carrera o hacerse de manera interdisciplinaria entre varias). Acorde a las tendencias que se manejan en todo el mundo, es posible agregar mas temas y materias de “*Ciencia de Datos*” a nuestras carreras y especialidades e incluso una especialidad completa en “*Ciencia de datos*”.

Es tal la importancia de la “*Ciencia de Datos*”, que universidades como la UNAM y el Instituto Politécnico Nacional de México, han optado por abrir carreras completas en Ciencia de Datos, es por ello que el Tecnológico Nacional de México debe apoyar todos los esfuerzos que se hagan para tener presencia en esta área tan importante de la “*Ciencia de Datos*” en nuestro país.

Dedicatoria

“Los datos os harán libres”

Rogelio Ferreira Escutia

Dedicado a todos aquellos que persiguen la verdad a través de la Ciencia y la Tecnología.

Agradecimiento

Agradezco al Profesor José Manuel Cuiñ Jacuinde (Jefe del Departamento de Sistemas y Computación) y al Profesor J. Guadalupe Ramos Díaz (Presidente de la Academia de Sistemas y Computación) por su valioso apoyo durante el trámite para ser aceptado, revisado y aprobado, éste Proyecto como parte de mi Año Sabático.

Agradezco a los Profesores del ITM:

Jesús Eduardo Alcaraz Chávez

Juan Carlos Olivares Rojas

Juan Jesús Ruíz Lagunas

Por sus comentarios y sugerencias que me hicieron durante la revisión del presente libro, para poder mejorar la versión final.

Agradezco al “*Tecnológico Nacional de México*”, al “*Instituto Tecnológico de Morelia*” y al “*Departamento de Sistemas y Computación*” por haberme permitido la realización de mi “*Año Sabático*” realizando este libro de “*Ciencia de Datos, Teoría y Aplicaciones*”, ya que ha sido como un increíble viaje por el mundo de la tecnología actual, que culmina con la terminación de este libro, y más que que un trabajo, ha sido un deleite haberlo realizado.

Este ha sido un “Libro elaborado durante el ejercicio del periodo Sabático autorizado por el TecNM”.

A todos aquellos que de alguna manera me ayudaron para la terminación del mismo...

¡Muchas Gracias!

Índice

Prólogo	17
Capítulo 1 - Introducción a Ciencia de Datos	
1.1 Definición y conceptos	21
1.1.1 ¿Qué es Ciencia de Datos?	21
1.1.2 Definición	21
1.1.3 ¿Qué es un científico de datos?	22
1.2 Historia	22
1.2.1 Los inicios	22
1.2.2 Surgimiento de la Ciencia de Datos	22
1.3 Habilidades de un Científico de Datos	23
1.3.1 Principales habilidades que debe tener un Científico de Datos	23
1.3.2 Habilidades Matemáticas y Estadísticas	23
1.3.3 Habilidades del área de Computación	23
1.3.4 Conocimiento del negocio	23
1.3.5 Retos de un Científico de Datos	24
1.4 Técnicas y Herramientas para Ciencia de Datos	24
1.4.1 Etapas a desarrollar en un proyecto de Ciencia de Datos	24
1.4.2 Lenguajes de programación	25
1.4.3 Entornos de programación	26
Capítulo 2 - Origen y Manejo de Datos	
2.1 Definición y conceptos de datos	27
2.1.1 Definición de datos	27
2.1.2 Definición de buenos y malos datos	28
2.2 Origen de los datos	28
2.2.1 Cantidad de datos generada	28
2.2.2 Origen de los datos	29
2.2.3 Tendencias de generación de datos	30
2.3 Formatos y extracción de datos	30
2.3.1 Tipos de datos	30
2.3.2 Archivos “txt”	31
2.3.3 Archivos “CSV”	32
2.3.4 Archivos “XLS”	34
2.3.5 Archivos “PDF”	35
2.3.6 Archivos “XML”	36
2.3.7 Archivos “JSON”	36
2.3.8 Archivos “HTML”	37
Capítulo 3 - Almacenamiento de Información	
3.1 Dispositivos de almacenamiento	39
3.1.1 Clasificación	39
3.2 Bases de datos	39
3.2.1 Bases de datos Pre-Relacionales	40

3.2.2 Bases de datos Relacionales	40
3.2.3 Bases de datos Post-Relacionales	42
3.3 Bases de datos distribuidas	43
3.4 Datacenters	43
3.5 Cómputo en la nube	45
3.5.1 Software como Servicio	45
3.5.2 Plataforma como Servicio	45
3.5.3 Infraestructura como Servicio	45
3.6 Virtualización	46
3.6.1 Arquitecturas de Virtualización	46
3.6.2 Virtualización sin un sistema operativo anfitrión	46
3.6.3 Virtualización a partir de un sistema operativo anfitrión	46
3.7 Microservicios	47
3.8 Contenedores	47
3.9 Control de Versiones	47
3.10 Blockchain	48
3.11 Dataset	49

Capítulo 4 - Estadística

4.1 Población y Muestra	51
4.1.1 Población	51
4.1.2 Muestra	52
4.2 Histogramas	52
4.3 Mínimo, Máximo y Rango	52
4.3.1 Mínimo y Máximo	52
4.3.2 Rango	53
4.4 Medidas de Tendencia Central	53
4.4.1 Media	53
4.4.2 Mediana	53
4.4.3 Moda	53
4.5 Medidas de Dispersión	54
4.5.1 Dispersión	54
4.5.2 Varianza	54
4.5.3 Desviación Estándar	55
4.6 Correlación	55
4.7 Regresión Lineal	55

Capítulo 5 - Probabilidad

5.1 Conceptos	61
5.1.1 Definición de Probabilidad	61
5.1.2 Propiedades básicas	62
5.1.3 Espacio de Muestreo	62
5.1.4 Ley de los grandes números	63
5.2 Espacio de Probabilidad	63
5.2.1 Definición	63
5.3 Probabilidad condicional	64
5.3.1 Definición	64
5.3.2 Probabilidad Condicional	65
5.3.3 Independencia	65
5.4 Variables Aleatorias	65
5.4.1 Definición	65
5.4.2 Variables Aleatorias Discretas	65

5.4.3 Variables Aleatorias Continuas	66
5.5 Distribuciones de Probabilidad	66
5.5.1 Definición	66
5.5.2 Distribución Binomial	66
5.5.3 Distribución de Poisson	66
5.5.4 Distribución Normal	67
5.5.5 Teorema del Límite Central	67
5.6 Modelos Gráficos Probabilísticos	68
5.6.1 Definición	68
5.6.2 Aplicaciones de los PGM	68
5.6.3 Clasificación de los PGM	68
5.6.4 Grafos No Dirigidos y Grafos Dirigidos	68
5.7 Redes Bayesianas	69
5.7.1 Definición	69
5.8 Modelos de Márkov	70
5.8.1 Definición	70

Capítulo 6 - Machine Learning

6.1 Conceptos	71
6.1.1 Definición	71
6.2 Historia	72
6.2.1 Primeros desarrollos	72
6.2.2 Primeras Aplicaciones	75
6.3 Clasificaciones de Algoritmos de Machine Learning	75
6.3.1 Aprendizaje Supervisado	75
6.3.2 Aprendizaje No Supervisado	77
6.3.3 Aprendizaje Semi-Supervisado	78
6.3.4 Aprendizaje por Refuerzo	78
6.4 Algoritmos de Machine Learning	78
6.4.1 Regresión Lineal	78
6.4.2 Regresión Logística	79
6.4.3 Árboles de Decisión	80
6.4.4 Clasificadores Probabilísticos	81
6.4.5 Algoritmos de Agrupamiento: K-Means	81
6.4.6 Support Vector Machines	82
6.4.7 KNN	82
6.4.8 Random Forest	83
6.4.9 Redes Neuronales	84
6.5 Empezando con Machine Learning	84
6.5.1 Datasets	84
6.5.2 Entrenamiento	85
6.5.3 Sobreajuste y Subajuste	85
6.6 Aplicaciones con Machine Learning	85
6.6.1 Regresión lineal	86

Capítulo 7 - Deep Learning

7.1 Conceptos	89
7.1.1 Definición	89
7.2 Historia	90
7.2.1 Primeros desarrollos	90
7.2.2 Primeras Aplicaciones	92
7.3 Clases de Deep Learning	92

7.4 Redes Neuronales	93
7.5 Redes Neuronales Profundas (Deep Learning)	94
7.6 Redes Neuronales Convolucionales	94
7.7 Redes Adversarias Generativas (GAN's)	94
7.8 Aprendizaje por Refuerzo Profundo	95
7.8.1 Conceptos sobre Aprendizaje por Refuerzo Profundo	95
7.8.2 Componentes del Aprendizaje por Refuerzo Profundo	95
7.9 Transformers	96
7.9.1 Conceptos	96
7.10 Tendencias	96
7.10.1 Neuroevolución	96
7.11 Aplicaciones con Deep Learning	97
7.11.1 Reconocimiento de Números	97

Capítulo 8 - Procesamiento del Lenguaje Natural

8.1 Conceptos Básicos	107
8.1.1 Definición	107
8.1.2 Aplicaciones	108
8.1.3 Lenguaje Humano	108
8.1.4 Historia	109
8.1.5 Herramientas de Desarrollo	110
8.2 Análisis de Textos	111
8.2.1 Corpus y Fuentes Léxicas	111
8.2.2 Análisis Estadístico	111
8.2.3 Limpieza de Textos	113
8.2.4 Segmentación de Textos	114
8.2.5 Palabras Importantes	118
8.2.6 Nubes de Palabras	120
8.2.7 Análisis de Sentimientos	123
8.2.8 Detección de Similaridades	130
8.3 Construcción de ChatBots para Lenguaje Natural	133
8.3.1 Conceptos sobre ChatBots	133
8.3.2 Aplicaciones de ChatBots	134
8.3.3 Ubicación de ChatBots	134
8.3.4 Clasificación de ChatBots	134

Capítulo 9 - Herramientas y Aplicaciones Avanzadas

9.1 Herramientas en la Nube	140
9.1.1 IBM Watson Studio	140
9.1.2 Google Cloud AutoML	141
9.1.3 Azure Machine Learning	143
9.1.4 Amazon Machine Learning	144
9.2 Aplicaciones Avanzadas	145
9.2.1 GPT	145
9.2.2 GitHub Copilot	146
9.2.3 DALL-E	147
9.2.4 Imagen	149
9.2.5 LaMDA	150
9.2.6 Stable Diffusion	151

Epílogo	153
Bibliografía	155
Anexos	
Anexo 1 – Tecnologías utilizadas	159
Anexo 2 - Instalación de Python	161
Anexo 3 - Instalación de Visual Studio Code	163
Anexo 4 - Instalación de bibliotecas en Python	165
Anexo 5 - Instalación de XAMPP	171
Anexo 6 - Instalación de un Cluster	175
Anexo 7 - Instalación de MongoDB	181
Anexo 8 - Instalación de una máquina virtual	185
Anexo 9 - Buenas prácticas de programación con Python	189
Anexo 10 - Lista de Datasets	191
Anexo 11 – Oficio de Sabático	193
Anexo 12 – Oficio de Sabático (Revisión)	195

Prólogo

La Ciencia de Datos

Es una ciencia interdisciplinaria de reciente creación que se encarga de la búsqueda, extracción y procesamiento de la información, por lo cual requiere nuevos perfiles profesionales que se adecúen a las necesidades que se avecinan con el creciente crecimiento exponencial de la información.

Pre-Requisitos

Aunque no es estrictamente necesario, pero para la lectura de este libro se recomienda tener conocimientos previos de 2 áreas importantes, las Matemáticas y la Computación. En Matemáticas se utilizarán los conceptos de Probabilidad y Estadística principalmente. En Computación se recomienda tener experiencia previa en programación, en especial en Python, aunque durante el transcurso del libro se explicarán los códigos y técnicas utilizadas, así como recomendaciones para mejorar la habilidad de programar.

Organización del libro

Este libro se compone de 9 capítulos, los cuales van desde la historia y surgimiento de la “*Ciencia de Datos*”, hasta llegar a los algoritmos y procesamientos más utilizados actualmente con Inteligencia Artificial, pasando por la búsqueda, la extracción, el procesamiento y la graficación de la información. Conforme se va avanzando en el libro se recomienda realizar las diferentes actividades prácticas para ir logrando un mejor aprovechamiento de cada tema, es por ello que se recomienda la lectura secuencial de este libro.

Convenciones usadas en este libro e Información Adicional

Identidad Gráfica

Para la realización de este libro se tomó en cuenta los lineamientos a seguir para la “*Imagen Institucional*” que ha determinado el “*Instituto Tecnológico de Morelia*” (ITM, 2021), así como el “*Manual de Identidad Gráfica*” establecido por el “*Tecnológico Nacional de México*” para todas sus publicaciones (TecNM, 2021).

Emojis

Se utilizaron los siguientes Emojis, con su correspondiente explicación:



Actividad Práctica



Códigos de Ejemplo



Buenas Prácticas



Material Adicional



Advertencia

Los Emojis fueron obtenidos de “Twemoji”, un proyecto Open Source de Twitter (Twitter, 2021).

Diagramas de Red

Las íconos para realizar los diagramas de red que están en este libro fueron obtenidos de "Network Topology Icons" (figura 1), los cuales son de uso libre y que son proporcionados por la compañía CISCO (Cisco, 2021).



Figura 1: Algunos íconos de la Simbología de Cisco

Créditos

Las siguientes imágenes y fotografías (en orden de aparición en el libro) fueron obtenidas de Wikimedia Commons (entre paréntesis su autor original según Wikimedia):

Morpheus	John Wilder Tukey (Paul Halmos)
Jeff Wu (C. F. Jeff Wu)	William Edwards Deming (Ateitomas)
Jeannette Wing (World Economic Forum)	Edmond de Goncourt (Nadar)
Función lineal (Magister Mathematicae)	Edward Bulwer-Lytton (Henry William Pickersgill)
Retrato de mujer con paraguas 01 (Dorieo)	1 Peso (AKS.9955)
Binomial distribution (Tayste)	Normal Distribution (Inductiveload)
Graphical model example (Rgiuly)	Examples of an Undirected Graph (RJE42)
Example of a Directed Graph (RJE42)	Nodos de una Red Bayesiana Simple (Kilom691)
Red Bayesiana Simple (Efren Vila Alfonso)	Cadena de Márkov (Joxemai4)
John McCarthy (null0)	Geoffrey Hinton (Steve Jurvetson)
Joseph Weizenbaum (Ulrich Hansen)	Alan Turing (PhotoColor)
ML as a subfield of AI (Yakoove)	Part of ML as subfield (Yakoove)
Alan Turing retrato (autor desconocido)	Walter Pitts (autor desconocido)
Arthur Lee Samuel (XI2085)	Frank Rosenblatt (autor anónimo)
SRI Shakey with callouts (SRI International)	Stanford Cart (Don DeBold)
Gerald Ford (Eldon Lytle)	Regresión Lineal (Amatulic)
Regresión Logística (Qef (talk))	Árboles de Decisión (Gilgoldm)
Redes Bayesianas (Kilom691)	K Means (Weston.pace)
Support Vector Machine (Elisfm)	KNN (Antti Ajanki AnAj)
Random Forest (Venkata Jagannath)	Gato (Cayambe)
Perro (Crusier)	Intel 8088 (Thomas Nguyen)
IBM Personal Computer (Ruben de Rijcke)	CRAY-1 (Clemens PFEIFFER)
Deep Blue (James the photographer)	Gary Kasparov (Owen Williams)
Geoffrey Hinton (Eviatar Bach)	Computadora Watson (Clockready)
Lee Sedol (Cyberoro ORO)	Datacenter Google (Da mocavi)
Red Neuronal (en:User:Cburnett)	Ian Goodfellow (foto propia)
"Paradise" (dominio público)	GAN deepfake white girl (bod Inga klang)
MNIST (Josef Steppan)	Función Sigmoide (Dnu72)
Función ReLU (Ringdongdang)	Joseph Weizenbaum (Ulrich Hansen)
Alan Turing (Thmahe)	Chomsky (Duncan Rawlinson)

Wikimedia Commons es una colección de archivos de uso libre bajo ciertas licencias como "Creative Commons" y "Fair Use" (Wikimedia, 2021).

Las fotografías de las siguientes personas:

Leo Breiman

fueron obtenidas de HandWiki, el cual es sitio de uso libre dedicado a la ciencia, la tecnología y la computación (HandWiki, 2021).

Los logotipos de las tecnologías utilizadas en este libro fueron obtenidas de Wikimedia Commons, la cual es una colección de archivos de uso libre bajo ciertas licencias como “*Creative Commons*” y “*Fair Use*” (Wikimedia, 2021). Algunos de los logotipos de las tecnologías utilizadas son los siguientes:



En el “Anexo 1” se pueden consultar todas las “*Tecnologías utilizadas*” durante el presente libro.

Sistemas Operativos Usados

Los sistemas operativos utilizados durante el libro son:

- Ubuntu 22.04 (Jammy Jellyfish).
- MacOS X 12.5.1 (Monterey).
- Windows 8.1



Código del Libro

Todo el código de los programas hechos en Python que se encuentran a lo largo de este libro, se pueden descargar para su consulta de manera totalmente gratuita y abierta de los siguientes enlaces:

- http://www.xumarhu.net/libro_01.htm
- http://sagitario.itmorelia.edu.mx/~rogelio/libro_01.htm

Acerca del Autor:



Rogelio Ferreira Escutia es "Ingeniero en Electrónica" por el "Instituto Tecnológico de Morelia" (1990 a 1995) y tiene una "Maestría en Ciencias de la Computación" en el "Instituto Tecnológico y de Estudios Superiores de Monterrey" (ITESM) Campus Morelos (1997-1998).

Desde 1999 y hasta la fecha es Profesor de tiempo completo en el "Instituto Tecnológico de Morelia", adscrito al "Departamento de Sistemas y Computación", como parte del "Tecnológico Nacional de México" (TecNM).

Formas de contacto:

Para contactar al autor se puede hacer por correo electrónico, página Web o vía redes sociales, los cuales son los siguientes:

Correo Institucional:



rogelio.fe@morelia.tecnm.mx

Correo Personal:



rogeplus@gmail.com

Página Web:



<http://sagitario.itmorelia.edu.mx/~rogelio/>

Página Web Personal:



<http://www.xumarhu.net/>

Grupo de Divulgación de Tecnología:



xumarhu.net

Cuenta de Divulgación de Tecnología:



@rogeplus

Canal de Divulgación de Tecnología:



<https://www.youtube.com/channel/UC0on88n3LwTKxJb8T09sGjg>

Perfil Profesional:



rogelioferreiraescutia

Capítulo

Introducción a la Ciencia de Datos

1

Objetivo: Conocer el surgimiento del área de Ciencia de Datos, herramientas y habilidades necesarias para el manejo y procesamiento de la información.

“Yo solo puedo enseñarte la puerta, tú tienes que caminar a través de ella...”

Morpheus, The Matrix



En este primer capítulo se abordarán los conceptos básicos sobre “Ciencia de Datos” para dar una panorama general de esta área, así como su definición, la historia, las habilidades que requiere un científico de datos y las herramientas necesarias para desarrollar su trabajo.

1.1 Definición y conceptos

1.1.1 ¿Qué es Ciencia de Datos?

Es tan nueva que algunas personas todavía no la aceptan como una “*Ciencia*”, pero el crecimiento exponencial de la información ha planteado la necesidad de su análisis y por lo tanto del nacimiento y crecimiento de esta “*Nueva Ciencia*”, que incluso fue catalogada por Thomas J. Davenport como “*el trabajo mas sexi del siglo XXI*” (Davenport, 2012).

1.1.2 Definición

Debido a que el área de “*Ciencia de Datos*” como tal es relativamente nueva, su definición ha ido cambiando y se puede definir de varias maneras, por lo tanto podemos tomar la definición que

propone Wikipedia, la cual es que *“La ciencia de datos es un campo interdisciplinario que involucra métodos científicos, procesos y sistemas para extraer conocimiento o un mejor entendimiento de datos en sus diferentes formas, ya sea estructurados o no estructurados, lo cual es una continuación de algunos campos de análisis de datos como la estadística, la minería de datos, el aprendizaje automático, y la analítica predictiva”* (Wiki_001, 2021).

Según Jeanette Wing, *“La Ciencia de datos es el estudio de extraer valor de los datos”* (Ananthaswamy, 2021).

1.1.3 ¿Qué es un científico de datos?

Hasta hace poco tiempo, el *“Científico de Datos”* no existía como tal, y por lo tanto su definición también es reciente y sigue sufriendo cambios. Cada autor plantea pequeñas variaciones a la definición y a las actividades que realiza un *“Científico de Datos”*. Una definición práctica es la que propone Favio Vázquez, la cual dice que *“Un Científico de Datos es una persona (¿o sistema?) encargada de analizar problemas de negocios/organizaciones ofreciendo una solución estructurada que comienza convirtiendo este problema en una pregunta válida y completa, luego usa programación y herramientas computacionales para desarrollar códigos que preparan , limpian y analizan los datos para crear modelos y responder la pregunta inicial”* (Vázquez, 2018).

1.2 Historia

1.2.1 Los inicios

El primer análisis de datos del que se tenga historia lo realizó John Graunt, el cual era un matemático y estadístico inglés, que en 1662 analizó la tasa de mortalidad de Londres para tratar de conocer el comportamiento y la propagación de la peste bubónica por la ciudad. A él se le atribuye la creación de la primera *“tabla de mortalidad”*, la cual expresaba la expectativa de vida por edad de la población (Wiki_0003, 2021).

En 1884, Herman Hollerith trabajaba en la oficina de patentes de Estados Unidos en Washington D.C., se le ocurrió la creación de una máquina que utilizaba tarjetas perforadas donde se podría codificar información, la cual construyó y patentó en 1889. En 1890 el gobierno de los Estados Unidos eligió la máquina de Hollerith para ser usada durante el censo, la cual permitía codificar información de manera más fácil para después ser procesada rápidamente. Esto resultó en que se pudo terminar el censo en 3 años (que normalmente tardaba 10 años), lo cual supuso un gran avance en la captura y procesamiento de información (Wiki_0004, 2021).

1.2.2 Surgimiento de la Ciencia de Datos

En 1962, John W. Tukey (figura 2) fué de los primeros científicos que definió el *“Análisis de Datos”* como los *“Procedimientos para analizar datos, técnicas para interpretar los resultados de dichos procedimientos, formas de planificar la recopilación de datos para hacer su análisis más fácil, más preciso o acertado, y toda la maquinaria y los resultados de las estadísticas matemáticas que se aplican al análisis de datos”*. Esto lo publicó en su artículo *“The Future of Data Analysis”* (Wiki_0002, 2021).

El Profesor Jeff Wu (figura 3), de la Universidad de Michigan, durante una conferencia planteó la pregunta de que si *¿“Estadística = Ciencia de Datos”?* y a partir de ahí, se popularizó el término *“Data Science”* (Wiki_0002, 2021).

En el 2001 un profesor de la Universidad de California en Berkeley Leo Breiman (figura 4) publicó el artículo "*Statistical Modeling: The two cultures*", donde comenta acerca de la estadística antigua y el surgimiento de una tipo de nueva estadística, donde se fusiona la estadística con las Ciencias Computacionales (Wiki_0005, 2021).



Figura 2: John Tukey

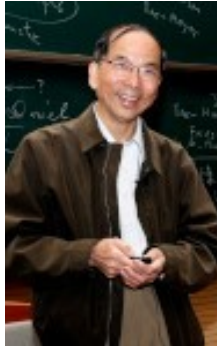


Figura 3: Jeff Wu



Figura 4: Leo Breiman

1.3 Habilidades de un Científico de Datos

1.3.1 Principales habilidades que debe tener un Científico de Datos

- 1) Habilidades Matemáticas y Estadísticas
- 2) Habilidades del área de Computación
- 3) Conocimiento del negocio

1.3.2 Habilidades Matemáticas y Estadísticas

- 1) Habilidades Matemáticas y Estadísticas
 - Matemáticas
 - Estadística
 - Modelado

1.3.3 Habilidades del área de Computación

- 2) Habilidades del área de Computación
 - Programación.
 - Redes
 - Seguridad
 - Bases de Datos
 - Sistemas Distribuidos

1.3.4 Conocimiento del negocio

- 3) Conocimiento del negocio
 - Clientes

- Ventas
- Artículos
- Mercados
- Marketing

1.3.5 Retos de un Científico de Datos

Algunos de los puntos a resolver durante un proyecto de Ciencia de Datos según Favio Vázquez (Vázquez, 2020) son:

- Entender el contexto del problema.
- Tener datos de calidad.
- Limpieza de datos.
- Preparación de la información.
- Seguridad de la información.
- Pruebas de calidad, unidad, integración y estrés.
- Integración continua.
- Despliegue de modelos
- Monitores de proyectos, métricas y modelos

De acuerdo a la encuesta de salarios que realizó la editorial O'Reilly (2021 Data/AI Salary Survey), el salario promedio de los profesionales que trabajan en el área de datos es de \$146,000 dólares americanos, el cual corresponde a uno de los más altos del área de tecnología (Loukides, 2021).

1.4 Técnicas y Herramientas para Ciencia de Datos

1.4.1 Etapas a desarrollar en un proyecto de Ciencia de Datos

Los pasos a seguir por un científico de datos son:

- 1) Definir Objetivo
- 2) Extracción de datos
- 3) Preparar los datos
- 4) Exploración de los datos
- 5) Modelado de los datos
- 6) Presentación de los datos

Estas etapas pueden variar de acuerdo en cuanto al número y nombre de cada una de ellas de acuerdo al autor que se esté citando, en este caso son las que enumera Davy Rielen en su libro (Rielen et al., 2016).

1) Definir Objetivo

- Especificar a donde se quiere llegar:
- Predecir resultados.
- Descubrir tendencias.
- Encontrar patrones.

2) Extracción de datos

- Localizar la ubicación de los datos para poder leerlos y en su caso almacenarlos en nuestro sistema de base de datos.

3) Preparar los datos

- Antes de hacer algún proceso, se hace un preprocesamiento o limpiado de los datos, con el objetivo de poder tener solamente los datos requeridos y en el formato deseado.

4) Exploración de los datos

- En este paso se analizan y procesan los datos.

5) Modelado de los datos

- A partir de los datos se crea un modelo para lograr llegar a nuestro objetivo y encontrar resultados.

6) Presentación de los datos

- Finalmente hay que visualizar (graficar) los datos obtenidos para poder entender más fácilmente los resultados.

1.4.2 Lenguajes de programación

Python, R, Julia, Scala y C++ son los lenguajes más utilizados en Ciencia de Datos. De los anteriores el más utilizada es Python, el cual, no solamente es un lenguaje de propósito general, si no que es el más utilizado para hacer análisis de datos de alto nivel y esto es debido a que se han incluido paquetes muy importantes como Pandas, SkLearn, TensorFlow, Keras, Numpy y muchos más (Boudreau, 2020).

De acuerdo a la encuesta de salarios que realizó la editorial O'Reilly (2021 Data/AI Salary Survey), Python es el lenguaje de programación más utilizado para el análisis de datos con un 61%, y en cuestión de salarios el saber Python se logra llegar a los \$150,000 dólares americanos en promedio por encima de otros lenguajes (Loukides, 2021).

Según la encuesta 2021 desarrollada por Stack OverFlow (Developer Survey 2021) Python se coloca como el lenguaje de uso general más popular con un 48.24% (Stack_0001, 2021).



Actividad: Instalación de Python

Descargar e instalar Python en nuestra computadora. En el “Anexo 2” se encuentra la “Instalación de Python”.



Buenas Prácticas: Para iniciar a programar en Python se recomienda seguir las guías oficiales (“*Style Guide for Python Code*”) el cual nos ayudará a generar un código más claro desde un principio y no generar algunos vicios de codificación más adelante. La Guía se puede consultar en: <https://www.python.org/dev/peps/pep-0008/>



Buenas Prácticas: Una guía adicional para “*Buenas prácticas de programación con Python*” se encuentra en el Anexo 9.



Advertencia: La Python Software Foundation anunció en el 2019 que se terminaría el soporte para Python 2 a partir de enero del 2020 (la última versión es la 2.7.18), por lo tanto se recomienda sólo usar versiones Python 3 en adelante (actualmente está la versión 3.10.6).



Material Adicional: Se recomienda leer el “*Zen of Python*”, el cual es una lista de principios para el lenguaje Python: <https://www.python.org/dev/peps/pep-0008/>
También se pueden consultar desde el prompt de Python tecleando: “import this”

1.4.3 Entornos de programación

Según la encuesta 2021 desarrollada por Stack OverFlow (Developer Survey 2021) Visual Studio Code (VSC) se coloca como el IDE más utilizado con 71.06% (Stack_0001, 2021).



Actividad: Instalación de VSC.

Descargar e instalar VSC en nuestra computadora. En el “Anexo 3” se encuentra la “*Instalación de Visual Studio Code*”.



Material Adicional: Para empezar a programar en Python (y no se tiene conocimiento previo) se recomienda este sitio donde se encuentra un tutorial gratuito:

<https://learnpython.org/>



Material Adicional: La documentación oficial de Python para su consulta se encuentra en:

<https://docs.python.org/3/tutorial/>

Capítulo

Origen y Manejo de Datos

2

Objetivo: Conocer el concepto de dato, características, origen y extracción de los datos.

“En Dios confiamos, todos los demás tienen que aportar datos”

William Edwards Deming

William Edwards Deming (1900-1993) fue un Profesor universitario y estadístico estadounidense. Su mayor aportación fue al desarrollo de los procesos de calidad en Japón. Los "14 Principios de Deming" se presentaron en el libro "Out of the Crisis" (Salir de la Crisis) que Deming escribió y que fueron la base para los grandes cambios en la industria norteamericana.



En este capítulo se definirá el concepto de “*datos*”, su definición así como su origen y la cantidad de datos que se están generando actualmente. También se explicarán los tipos de datos más usados dentro del área de “*Ciencia de Datos*” así como ejemplos de su formato y forma de extracción para su lectura y análisis.

Esta unidad es muy importante, ya que durante todo el libro se estará haciendo referencia a los tipos de datos que a continuación se detallarán.

2.1 Definición y conceptos de datos

2.1.1 Definición de dato

Según Harry F. Foxwell, “*Dato*” es un “*una colección de símbolos que representan un conjunto de medidas u observaciones acerca de un evento*” (Foxwell, 2020).

2.1.2 Definición de buenos y malos datos

Características de un “*buen*” dato (Foxwell, 2020):

- Exacto.
- Relevante.
- Representativo.
- Bien definido.
- Completo.
- Granular.

Causas de un “*mal*” Dato (Foxwell, 2020)::

- Dato mal definido (confuso).
- Tipo de dato no especificado.
- Tipo de dato que no se ajusta a los estándares.
- Mala recolección o grabación de datos.
- Mala interpretación de los datos.
- Error de copia ó transcripción de datos.
- Datos fraudulentos.
- Datos perdidos.
- Datos fuera de rango.
- Error de almacenamiento digital.
- Agregar ceros a los datos (0014 en vez de 14).
- Colocar la letra "o" en vez del número "0" (cero).
- Transposición de caracteres (LA por AL).
- Inconsistencia de los datos (MEX ó MX).

Según un estudio realizado por Tadhg Nagle, solamente el 3% de las compañías cumplen con los estándares básicos en la calidad de datos (Nagle et al., 2017).

2.2 Origen de los datos

2.2.1 Cantidad de datos generada

Es difícil calcular la cantidad de datos que se generan en Internet en tiempo real, pero varias empresas han hecho cálculos aproximados de acuerdo al tráfico de información que monitorean. En la figura 5 se encuentra una imagen generada a partir de la cantidad de información que se genera en Internet en un minuto según el sitio Statista (Jenik, 2021).

El auge actual de la ciencia de datos, precisamente es debido a la gran cantidad de datos que se están generando y que se requiere analizar.

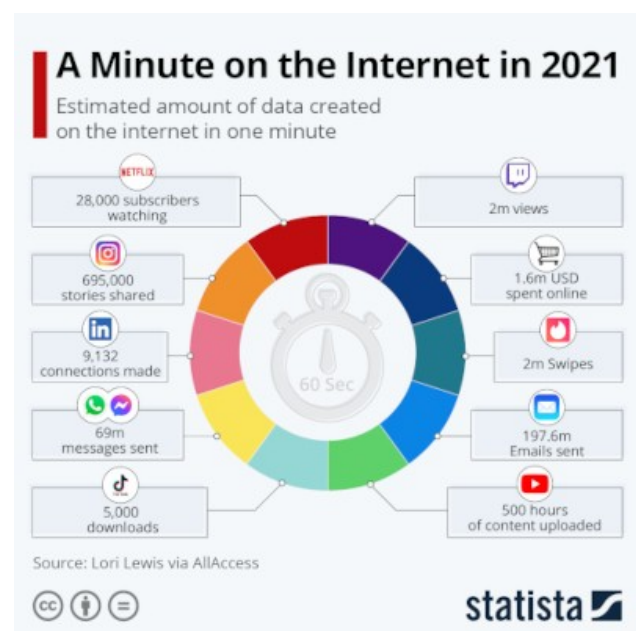


Figura 5: Cantidad de datos generada en 1 minuto

2.2.2 Origen de los datos

Los datos los podemos clasificar principalmente en 2, de acuerdo a su origen de generación:

- Internos
- Externos

Internos: Son todos aquellos que generamos en nuestra empresa o lugar de trabajo, así como los que generamos nosotros como usuarios finales. Son pequeños en comparación con lo que se genera en todo el Internet, pero es la base de los datos con que se alimenta el Internet.

Externos: Son todos aquellos datos que se generan en todo el mundo y que se publican en Internet o a través de él, por lo tanto son los que mas se generan y analizan.

De acuerdo a la cantidad de datos que se genera actualmente en Internet, podemos hacer una listado del origen de la información de acuerdo a cantidad de datos generados, de menor a mayor:

- Páginas estáticas
- Páginas de servicios
- Redes sociales
- Video en streaming
- Videoconferencias
- Internet de las Cosas

Páginas estáticas: Son aquellas con las que inicialmente arrancó la Web (llamada posteriormente la Web 1.0), y que consiste en páginas que no cambian su contenido, son generalmente de tipo informativo y en tamaño no han crecido mucho como las que vienen a continuación.

Páginas de servicios: Son sitios que interactúan con el usuario y ofrecen información en tiempo real como el clima, noticias, resultados deportivos, etc., así como también ofrecer información de procesos que están ocurriendo en una empresa (flujo de dinero, capital, intereses) ó en una industria para monitorear procesos (temperatura, presión, humedad). Estos sitios están alimentados por bases de datos que se actualizan constantemente (algunos en tiempo real) y que por lo tanto pueden generar gran cantidad de información conforme avanza el tiempo.

Redes sociales: Estos sitios crecieron rápidamente, principalmente desde la aparición de Facebook ya que lograron atraer una gran cantidad de usuarios que a su vez se encargaron de llenar de contenido propio, logrando en conjunto una gran acumulación de datos. Actualmente otras redes de reciente creación como Instagram y TikTok se han encargado de aumentar la cantidad de datos generada en las redes sociales.

Video en streaming: Estos servicios se encargan de enviar video sobre demanda a los usuarios, principalmente películas y series. YouTube fue el primero en iniciar la era del video en streaming, aunque inicialmente estaba muy restringido por el poco ancho de banda disponible sobre Internet, lo cual lo hacía lento. Posteriormente años después, Netflix se encargó de popularizar este medio, el cual ya era más rápido gracias a los avances en la interconexión de clientes y servidores con una baja latencia en el servicio. Actualmente siguen surgiendo nuevas empresas que ofrecen estos servicios cada vez a mayor calidad y por lo tanto gastando mas ancho de banda para el envío de información.

Videoconferencias: A partir del inicio de la Pandemia y del confinamiento mundial generado por la misma, muchos de los trabajos se empezaron a hacer en línea, por lo cual los servicios de

videoconferencia empezaron a crecer rápidamente para poder ofrecer conexión entre personas y equipos de trabajo. Actualmente sigue en crecimiento y también genera una gran cantidad de datos.

Internet de las Cosas: Otra fuente importante del origen de los datos son los dispositivos independientes que generan datos y los pueden enviar por Internet a los servidores, a esto se le llama “*Internet de las Cosas*” ó más comúnmente y para abreviar “*IoT*” (“*Internet of Things*” por su significado en inglés). Estos dispositivos pequeños y baratos pueden colocarse en muchos lugares de donde pueden obtener datos a través de sensores y cámaras que pueden generar información como puede ser temperatura, presión, movimiento, ubicación, video, sonido, etc.. Estos dispositivos pueden ser nuestros teléfonos, relojes, cámaras de vigilancia y cada día aparecen en mas lugares, generando en tiempo real una gran cantidad de datos.

2.2.3 Tendencias de generación de datos

Es difícil predecir el futuro con certeza, pero al menos actualmente se observan principalmente 4 fuentes principales de generación de datos que están en crecimiento y que seguirán llenando de información el Internet y que son los últimos 4 de la lista anterior:

- Redes sociales.
- Video en streaming.
- Videoconferencias.
- Internet de las Cosas.

Según la cantidad de datos que se generan por minuto de acuerdo al sitio Statista (Jenik, 2021), se pueden hacer cálculos aproximados de que la información total que se encuentra en Internet se estará duplicando cada 2 años. Por lo anterior, la demanda de análisis de datos y por consiguiente, de científicos de datos aumentará bastante, por lo que es necesario una mayor investigación y capacitación de profesionistas en esta área, que seguramente será la de mas demanda en los próximos años, y que actualmente ya es de las de mayor crecimiento.

2.3 Formatos y extracción de datos

2.3.1 Tipos de datos

Tipos de Datos (según su estructura):

- Estructurados.
- Semi-estructurados.
- No Estructurados.

Tipos de Datos (según su ubicación):

- Local: HD, SSD, SD, etc.
- En Red: Computadoras cercanas.
- Internet: Servidores, páginas Web, Datasets.

Tipos de Datos (según su formato):

- Texto (extensión .txt)
- CSV
- XLS
- PDF
- XML

- JSON
- HTML

Tipos de Datos (según su contenido):

- Nominales: Atributos cualitativos:
- Ordinales: Atributos cuantitativos pero no comparables.
- Proporcional: Cantidades medibles.
- Intervalo: Cantidades medibles (sin haber un cero).

2.3.2 Archivos "txt"

“Un archivo de texto simple, texto sencillo o texto sin formato (también llamado texto plano o texto simple; en inglés *“plain text”*), es un archivo informático que contiene únicamente texto formado solo por caracteres que son legibles por humanos, careciendo de cualquier tipo de formato tipográfico (Wiki_0007, 2021).

Dentro de los archivos "txt", podemos encontrar 2 tipos de archivos, de acuerdo a la forma en que fueron grabados, los cuales son:

- Archivos de Texto, también denominados *“texto plano”* (figura 6).
- Archivos Binarios (figura 7).

Los archivos de texto son los que contiene caracteres en texto plano. Normalmente son legibles al abrirlas con cualquier editor de texto (ejemplo: los archivos README).

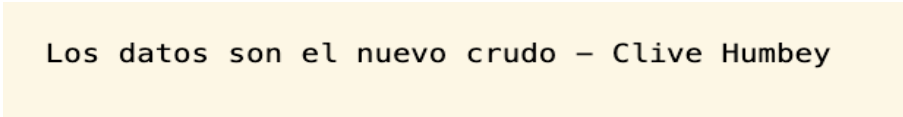


Figura 6: Ejemplo de un archivo de texto plano

Los archivos binarios están almacenados por bytes y no son fáciles de leer cuando se abren en un editor de texto común, es necesario tener la aplicación adecuada que reconozca el formato.



Figura 7: Ejemplo de un archivo binario al abrirlo en un editor de texto

Para abrir un archivo se utiliza la instrucción "open", luego el nombre del archivo y su extensión. Existe un parámetro opcional (mode) que depende de la operación que se esté realizando.

Sus parámetros de operación son los siguientes:

r (Read): Para abrir un archivo sin modificarlo.

r+ (Read/Write): Para abrir y leer.

a (Append): Agregar datos al final de un archivo sin modificar su contenido.

W (Write): Para abrir y escribir un archivo.

X (Create): Para crear archivos.



Actividad: Lectura de un archivo “txt”

Leer el archivo “*readme.txt*” e imprimir su contenido.

Nota: El archivo “*readme.txt*” es un archivo de texto plano que hay que crear previamente.

```
1 archivo_externo = "readme.txt"
2 archivo_en_memoria = open(archivo_externo)
3 contenido = archivo_en_memoria.read()
4 print("\nEl contenido del archivo: " + archivo_externo + " es:\n")
5 print(contenido + "\n")
6 archivo_en_memoria.close()
```

El contenido del archivo: *readme.txt* es:

Los datos son el nuevo crudo - Clive Humbey

Otra forma de leer un archivo es en “*bloque*”, sin necesidad de cerrar la lectura del archivo como se muestra a continuación (la salida es exactamente la misma):

```
1 archivo_externo = "readme.txt"
2 with open(archivo_externo) as archivo_en_memoria:
3     contenido = archivo_en_memoria.read()
4     print("\nEl contenido del archivo: " + archivo_externo + " es:\n")
5     print(contenido + "\n")
```

Otros archivos pueden estar codificados en “*Unicode*” de un tipo específico, y para poder leerlos se requiere especificar la codificación del archivo como se muestra a continuación:



Actividad: Leer un archivo de texto externo que está en Unicode

Leer el archivo “*readme.txt*” e imprimir su contenido.

Nota: El archivo “*readme.txt*” es un archivo de texto plano que hay que crear previamente.

```
1 archivo_externo = "readme.txt"
2 with open(archivo_externo, "r", encoding="utf-8") as archivo_en_memoria:
3     contenido = archivo_en_memoria.read()
4     print("\nEl contenido del archivo: " + archivo_externo + " es:\n")
5     print(contenido + "\n")
```

2.3.3 Archivos “CSV”

Los archivos separados por comas, ó comúnmente llamados CSV (“*comma-separated values*”, por sus siglas en inglés) son un tipo de documento en formato abierto sencillo para representar datos en

forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal como en Chile, Perú, Argentina, España, Brasil, entre otros) y las filas por saltos de línea” (Wiki_0008, 2021).

Son aquellos archivos de texto, donde la información se almacena por partes (como en una base de datos o en una hoja de cálculo) y cada parte (que pueden ser nombres, direcciones, edades, etc.) se separa por comas y por renglones para su fácil lectura (figura 8).

Año	Marca	Modelo	Descripción	Precio
1997	Ford	E350	ac, ABS, moon	3000.00
1999	Chevy	Venture	Extended Edition	4900.00
1999	Chevyr	Venture	Extended Edition, Very Large	5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

Año,Marca,Modelo,Descripción,Precio
 1997,Ford,E350,"ac, abs, moon",3000.00
 1999,Chevyr,Venture,Extended Edition,4900.00
 1999,Chevy,Venture,"Extended Edition, Very Large",5000.00
 1996,Jeep,Grand Cherokee,"MUST SELL! air, moon roof, loaded",4799.00

Figura 8: Ejemplo de codificación de un CSV



Actividad: Leer un archivo de texto externo por renglones en formato CSV

Leer el archivo “*readme_2.txt*” e imprimir su contenido

Nota: El archivo “*readme_2.txt*” es un archivo de texto plano que hay que crear previamente, para este ejemplo es una archivo donde el primer renglón es una ciudad (Morelia) y el segundo es su temperatura (24).

```

1 archivo_externo = "readme_2.txt"
2 with open(archivo_externo) as archivo_en_memoria:
3     print("\nEl contenido del archivo: " + archivo_externo + " es:")
4     primer_renglon = archivo_en_memoria.readline()
5     print("\nPrimer renglón: " + primer_renglon)
6     segundo_renglon = archivo_en_memoria.readline()
7     print("Segundo renglón: " + segundo_renglon + "\n")

```

```

El contenido del archivo: readme_2.txt es:
Primer renglón: Morelia
Segundo renglón: 24

```

La forma mas utilizada de procesar información en memoria, es transferir los datos que se acaban de leer de un archivo y pasarlos a una lista para su posterior procesamiento



Actividad: Leer un archivo de texto externo por renglones en formato CSV y pasarlo a una lista

```

1 archivo_externo = "readme_2.txt"
2 with open(archivo_externo) as archivo_en_memoria:
3     print("\nEl contenido del archivo: " + archivo_externo + " es:")
4     contenido = archivo_en_memoria.readlines()
5     print(contenido)

```

El contenido del archivo: readme_2.txt es:
['Morelia\n', '24']

2.3.4 Archivos “XLS”

Es el formato que usa para almacenar la información de las hojas de cálculo de Microsoft Excel (figura 9). La primera versión salió para computadoras Mac en 1985 y actualmente hay versiones para varias plataformas y se ha convertido en uno de los formatos mas utilizados para el almacenamiento de información numérica en computadoras de escritorio.



Actividad: Leer un archivo XLS e imprimir el número de filas,

de columnas y el contenido.
Nota: El archivo XLS es una lista de calificaciones de prácticas de un grupo de 10 alumnos. Para este código se necesita instalar “pandas” y “xlrd”. En el Anexo 4 se encuentra la “Instalación de bibliotecas en Python”.

	A	B	C	D	E	F	G	H	I	J	K
1	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	PdePrac
2	100	100	100	90	100	100	100	100	90	80	96
3	100	100	85	100	100	100	100	100	95	70	95
4	100	100	100	90	100	100	100	100	90	100	98
5	100	100	70	100	100	100	100	100	100	75	94.5
6	100	100	70	100	100	100	100	100	100	85	95.5
7	100	100	90	100	100	100	100	100	100	70	96
8	100	100	100	100	80	100	100	100	90	97	96.7
9	100	100	85	100	100	100	100	100	95	98	97.8
10	100	100	0	80	100	100	90	100	90	82	84.2
11	90	80	70	86	94	81	98	100	97	78	87.4
12	99	98	77	95	97	98	99	100	95	84	94.11

Figura 9: Ejemplo de una hoja de cálculo

```
1 # Se requiere "pandas" > pip3 install pandas
2 # Se requiere "xlrd" > pip3 install xlrd
3 # Cargar la biblioteca de "pandas"
4 import pandas as pd
5 # cargar la hoja de excel
6 hoja_excel = pd.read_excel("excel_hoja_01_calificaciones.xls")
7 # Obtener las filas
8 filas = hoja_excel.shape[0]
9 print("\nFilas: " + str(filas))
10 # Obtener columnas
11 columnas = hoja_excel.shape[1]
12 print("Columnas: " + str(columnas))
13 # Imprimir contenido
14 print("\n", hoja_excel, "\n")
```

Filas: 11
Columnas: 11

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	PdePrac
0	100	100	100	90.0	100.0	100.0	100.0	100	90.0	80.0	96.00
1	100	100	85	100.0	100.0	100.0	100.0	100	95.0	70.0	95.00
2	100	100	100	90.0	100.0	100.0	100.0	100	90.0	100.0	98.00
3	100	100	70	100.0	100.0	100.0	100.0	100	100.0	75.0	94.50
4	100	100	70	100.0	100.0	100.0	100.0	100	100.0	85.0	95.50
5	100	100	90	100.0	100.0	100.0	100.0	100	100.0	70.0	96.00
6	100	100	100	100.0	80.0	100.0	100.0	100	90.0	97.0	96.70
7	100	100	85	100.0	100.0	100.0	100.0	100	95.0	98.0	97.80
8	100	100	0	80.0	100.0	100.0	90.0	100	90.0	82.0	84.20
9	90	80	70	86.0	94.0	81.0	98.0	100	97.0	78.0	87.40
10	99	98	77	94.6	97.4	98.1	98.8	100	94.7	83.5	94.11



Actividad: Leer una hoja de cálculo en formato de Excel y calcular el promedio de la primera fila

Nota: el archivo XLS es una lista de calificaciones de prácticas de un grupo de 10 alumnos y hay que calcular el promedio del primer alumno (fila 1, índice 0):

```
1 # Se requiere "pandas" > pip3 install pandas
2 # Se requiere "xlrd" > pip3 install xlrd
3 # Cargar la biblioteca de "pandas"
4 import pandas as pd
5 # cargar la hoja de excel en un dataframe y seleccionar la Hoja1
6 df = pd.read_excel("excel_hoja_01_calificaciones.xls", "Hoja1")
7 # Calcular el promedio de todas las filas
8 promedio = df.mean(axis=1)
9 # Imprimir el promedio de la primer fila
10 print("\nEl promedio de la primer fila es:", promedio[0], "\n")
```

El promedio de la primer fila es: 96.0

2.3.5 Archivos "PDF"

"PDF" (siglas en inglés de "Portable Document Format", "Formato de Documento Portable") es un formato de almacenamiento para documentos digitales independiente de plataformas de software o hardware (figura 10). Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto)" (Wiki_0009, 2021).

#	Nº Control	Nombre del Alumno	Tipo Evaluación
1	17120148	Becerra Constantino Yadira Araceli	Curso normal
2	17120151	Castro Cazares Carlos Jahir	Curso normal
3	17120157	De Jesus Galvan Brian Fernando	Curso normal

Figura 10: Ejemplo de codificación de un archivo PDF

2.3.6 Archivos “XML”

“XML”, siglas en inglés de “eXtensible Markup Language”, traducido como “Lenguaje de Mercado Extensible” o “Lenguaje de Marcas Extensible” (figura 11). Es un metalenguaje que permite definir lenguajes de marcas desarrollado por el “World Wide Web Consortium” (W3C) utilizado para almacenar datos en forma legible (Wiki_0010, 2021).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
  <Mensaje>
    <Remitente>
      <Nombre>Nombre del remitente</Nombre>
      <Mail> Correo del remitente </Mail>
    </Remitente>
    <Destinatario>
      <Nombre>Nombre del destinatario</Nombre>
      <Mail>Correo del destinatario</Mail>
    </Destinatario>
    <Texto>
      <Asunto>
        Este es mi documento con una estructura muy sencill
        no contiene atributos ni entidades...
      </Asunto>
      <Parrafo>
        Este es mi documento con una estructura muy sencill
        no contiene atributos ni entidades...
      </Parrafo>
    </Texto>
  </Mensaje>
</Edit_Mensaje>
```

Figura 11: Ejemplo de un formato XML

2.3.7 Archivos “JSON”

“JSON” (acrónimo de “JavaScript Object Notation”, “Notación de Objeto de JavaScript”) es un formato de texto sencillo para el intercambio de datos y se trata de un subconjunto de la notación literal de objetos de JavaScript (figura 12). Debido a su amplia adopción como alternativa a XML, se le considera (año 2019) un formato independiente del lenguaje” (Wiki_0011, 2021).

```
{
  "departamento":8,
  "nombredepto":"Ventas",
  "director": "Juan Rodríguez",
  "empleados": [
    {
      "nombre":"Pedro",
      "apellido":"Fernández"
    },{
      "nombre":"Jacinto",
      "apellido":"Benavente"
    }
  ]
}
```

Figura 12: Ejemplo de codificación JSON

2.3.8 Archivos “HTML”

“HTML”, siglas en inglés de “HyperText Markup Language” (“Lenguaje de Marcado de Hipertexto”), hace referencia al lenguaje de marcado para la elaboración de páginas web (figura 13). Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros (Wiki_0012, 2021).

Bienvenido al mundo del Web Scraping!

Esta es una página de pruebas sobre Web Scraping

Web Scraping es un conjunto de técnicas para extraer información de páginas Web.

Login:

Password:

Enlaces de Prueba

[Xumarhu](#)

[Tec de Morelia](#)

[Departamento de Sistemas y Computación](#)

```
<!DOCTYPE html>
<!-- Rogelio Ferreir Escutia - 10/marzo/2020 - Pruebas de Web Scraping -->
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Web Scraping</title>
    <meta name="keywords" content="web, scraping, búsquedas" />
    <meta name="description" content="Página para hacer pruebas de Web Scraping" />
    <meta name="author" content="Rogelio Ferreira Escutia" />
    <link rel="icon" type="image/png" href="favicon.ico" />
    <link rel="stylesheet" type="text/css" href="scraping.css" />
  </head>
  <body>
    <h1>Bienvenido al mundo del Web Scraping!</h1>
    <h2>Esta es una página de pruebas sobre Web Scraping</h2>
    <p>Web Scraping es un conjunto de técnicas para extraer información de páginas Web.</p>
    <form action="tiendita_login.php" method="post">
      <br /><input type="text" name="login"/><br/>
      <br /><input type="password" name="password"/><br/><br />
      <br /><input type="submit" value="Enviar"/>
    </form>
    <h2>Enlaces de Prueba</h2>
    <a href="http://www.xumarhu.net/">Xumarhu</a>
    <br /><br /><a href="http://www.itmorelia.edu.mx/">Tec de Morelia</a>
    <br /><br /><a href="http://dsc.itmorelia.edu.mx/">Departamento de Sistemas y Computación</a>
  </body>
</html>
```

Figura 13: Ejemplo de codificación de una página HTML

Otro lugar muy utilizado para extraer información son las páginas Web, las cuales están construidas básicamente por códigos HTML.



BeautifulSoup

Actividad: Extraer el título de una página Web.

Nota: Se requiere instalar BeautifulSoup.

```
1 from urllib.request import urlopen
2 from bs4 import BeautifulSoup
3 print("\nExtraer el título de una página Web")
4 pagina = "http://sagitario.itmorelia.edu.mx/~rogelio/"
5 html = urlopen(pagina)
6 bs = BeautifulSoup(html.read(), 'html.parser')
7 titulo = str(bs.title)
8 print("\nPágina: " + pagina)
9 print("Título: " + titulo+"\n")
```

Extraer el título de una página Web

Página: http://sagitario.itmorelia.edu.mx/~rogelio/
Título: <title>xumarhu.net - Portal de Tecnología</title>

Capítulo

Almacenamiento de Información

3

Objetivo: Conocer los diferentes tipos y formas de almacenamiento de datos.

“Es hora de que empecemos a fijarnos en los datos para resolver nuestros problemas”

Jeannette Wing

Jeannette Marie Wing (1956) es una ingeniera Estadounidense que estudió Informática en el MIT. Trabajó en la Universidad Carnegie Mellon y después fue vicepresidenta corporativa de Microsoft Research. Sus principales aportes han sido en el área de "pensamiento computacional".



Este capítulo trata acerca de las diferentes técnicas y métodos de almacenamiento de datos que se utilizan, desde las básicas hasta las más complejas. Cada una de ellas supone ventajas y desventajas, y cuando un científico de datos trabaja en la solución de un problema, puede encontrarse con algunas de ellas, por lo que es necesario conocerlas y poderlas implementar adecuadamente tanto para su lectura como el almacenamiento de la información.

3.1 Dispositivos de almacenamiento

3.1 Clasificación

De acuerdo al formato físico de almacenamiento algunos de los medios de almacenamiento más utilizados los podemos clasificar en:

- Magnético: Cinta magnética, Discos flexibles, discos duros.
- Ópticos: CDs, DVDs.
- Estado Sólido: Memorias Flash.

3.2 Bases de datos

“Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.” (Wiki_0013, 2021).

Las bases de datos han tenido varias etapas y según Guy Harrison existen principalmente 3 (Harrison, 2015):

- Pre-Relacionales.
- Relacionales.
- Siguiete Generación.

3.2.1 Bases de datos Pre-Relacionales

Esta primer generación de bases de datos abarca aproximadamente de 1950 a 1972 y se caracterizaba por almacenar las bases de datos en los primeros sistemas de almacenamiento que se empezaron a utilizar, donde el formato era básicamente lo que conocemos ahora como un archivo de texto plano, sin formato. Para acceder a la información se hacía un programa para obtener la información (figura 14).

Aunque esta forma es la mas antigua, sigue siendo la de mayor rendimiento y por lo tanto velocidad, es por eso que muchas aplicaciones actuales de ciencia de datos extraen información de archivos de texto, como los que están en formato CSV y JSON.

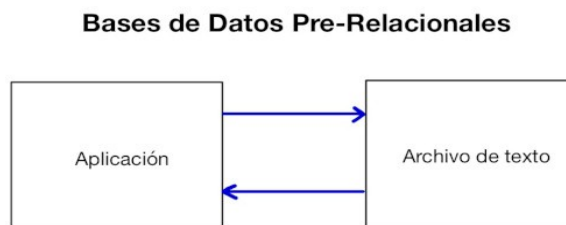


Figura 14: Bases de Datos Pre-Relacionales

3.2.2 Bases de datos Relacionales

Esta generación abarca aproximadamente de 1972 al 2005 y es donde la información se almacena en forma de tablas que contienen columnas y registros. Para encontrar información entre las diferentes tablas se crean las “*relaciones*” entre ellas (de ahí su nombre de “*relacional*”) y de esa manera se logra encontrar información que está ligada entre sí (figura 15).

La idea principal surge en 1970, cuando el inglés Edgar Frank Codd publica el artículo “*A Relational Model of Data for Large Shared Data Banks*” en donde publica los conceptos para el manejo de datos usando relaciones entre ellos (Codd, 1970). Este artículo se convirtió en la base para la creación de los “Modelos Relacionales” e inició la segunda generación de bases de datos.

En esta época se crean también los “*Manejadores de Bases de Datos*” ó “*DBMS*” (“*Data Base Management System*” por sus siglas en inglés), los cuales son programas que se encargan de interactuar por un lado con los usuarios y por el otro lado con la base de datos, haciendo más fácil la administración del sistema. En esta etapa también surgió el “*Lenguaje para Consultas Estructurado*” ó “*SQL*” (“*Structured Query Language*” por sus siglas en inglés), el cual se convirtió en el lenguaje estándar para la comunicación entre las aplicaciones con el DBMS, y hasta la fecha sigue siendo el mas utilizado.

Bases de Datos Relacionales

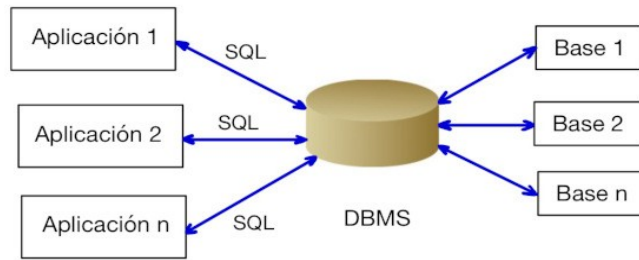


Figura 15: Bases de Datos Relacionales

Según la encuesta 2021 desarrollada por Stack OverFlow (Developer Survey 2021) MySQL se coloca como el sistema de Base de Datos mas utilizado con el 50.18% (Stack_0001, 2021).



Actividad: Instalación de XAMPP

Nota: En el “Anexo 5” se encuentra la “Instalación de XAMPP”, el cual es un sistema que instala Apache+PHP+MySQL.



Actividad: Conectarse a MySQL e imprimir el contenido de una tabla.

Nota: Se requiere instalar el conector ODBC entre Python y MySQL (MYSQL-CONNECTOR-PYTHON). Se requiere también ejecutar el siguiente script dentro de MySQL para crear la base, después crear la tabla e insertar registros en ella:

```
create database tiendita_catalogo;
use tiendita_catalogo;
create table catalogo (articulo text, precio float, cantidad int, imagen text);
insert into catalogo values("coca",6.50, 10, "coca.jpg");
insert into catalogo values("pepsi",5.50, 20, "pepsi.jpg");
```

Ahora hacer el programa en Python que leerá los registros que se acaban de insertar:

```
1 import mysql.connector as mysql      ## Se requiere importar para conectarse a MySQL
2 print ("Python conectándose a MySQL")
3 # Conectándose a MySQL
4 conexion = mysql.connect( host='localhost', user= 'root', passwd='', db='tiendita_catalogo' )
5 operacion = conexion.cursor()
6 operacion.execute( "SELECT * FROM catalogo" )
7 for articulo, precio, cantidad, imagen in operacion.fetchall() :
8     | print (articulo, precio, cantidad, imagen)
9     | conexion.close()
```



Actividad: Usando SQLite se crea una base, una tabla, se insertan 2 registros, se modifica un registro y luego se elimina. **Nota:** Hacer un programa en Python que haga lo anterior.

```

1 import sqlite3      # Importar SQLite versión 3
2 print('\n*** SQLite ***')
3 print('\nAbriendo la base de datos "informacion.db"') # Crear una conexión con una base de datos
4 conexion = sqlite3.connect('informacion.db')
5 c = conexion.cursor() # Crear un objeto cursor para referenciarlo
6 print('Creando la tabla: "agenda.db"') # Creando una tabla
7 c.execute("CREATE TABLE agenda (id integer PRIMARY KEY, nombre text, telefono text, edad int, peso real)")
8 print('Insertando un registro en la tabla: "agenda.db"') # Insertando 2 registros en la tabla
9 c.execute("INSERT INTO agenda VALUES(1, 'Juan', '443-312-15-70', 18, 74.5)")
10 c.execute("INSERT INTO agenda VALUES(2, 'Ana', '443-845-93-34', 25, 63.8)")
11 conexion.commit() # Guardando las operaciones realizadas
12 print('Leyendo todos los registros de la tabla:') # Leyendo todos los registros
13 c.execute('SELECT * FROM agenda')
14 print('Imprimiendo todos los registros de la tabla:') # Ciclo para imprimir todos los registros
15 registros = c.fetchall()
16 for ciclo in registros:
17     print(ciclo)
18 c.execute('UPDATE agenda SET edad = 45 where nombre = "Juan"') # Modificando un registro
19 conexion.commit()
20 print('Imprimiendo los registros ya modificados:') # Ciclo para imprimir
21 c.execute('SELECT * FROM agenda')
22 registros = c.fetchall()
23 for ciclo in registros:
24     print(ciclo)
25 c.execute('DELETE FROM agenda WHERE nombre = "Juan"') # Eliminando un registro
26 conexion.commit()
27 print('Imprimiendo los registros después de eliminar uno:') # Ciclo para imprimir
28 c.execute('SELECT * FROM agenda')
29 registros = c.fetchall()
30 for ciclo in registros:
31     print(ciclo)
32 print("\n")

```

```

*** SQLite ***

Abriendo la base de datos "informacion.db"
Creando la tabla: "agenda.db"
Insertando un registro en la tabla: "agenda.db"
Leyendo todos los registros de la tabla:
Imprimiendo todos los registros de la tabla:
(1, 'Juan', '443-312-15-70', 18, 74.5)
(2, 'Ana', '443-845-93-34', 25, 63.8)
Imprimiendo los registros ya modificados:
(1, 'Juan', '443-312-15-70', 45, 74.5)
(2, 'Ana', '443-845-93-34', 25, 63.8)
Imprimiendo los registros después de eliminar uno:
(2, 'Ana', '443-845-93-34', 25, 63.8)

```

3.2.3 Bases de datos Post-Relacionales

La generación actual llamada Post-Relacional, surge aproximadamente en 2005 y persiste hasta la actualidad. Esta generación se caracteriza por el gran incremento de información que se empezó a generar y que llevó al límite a los sistemas relacionales, por lo que se buscaron formas más eficientes para almacenar y recuperar información, iniciándose la tercera generación de bases de datos.

Por lo anterior, surgen los sistemas “NoSQL”, los cuales tienen las siguientes características

- No usan estructuras fijas como las tablas, si no mas bien archivos de texto.
- No hacen una relación directa entre los datos (no tienen soporte para hacer un “join”).

- No usan el sistema SQL como su lenguaje principal de consultas.
- No garantizan el concepto “ACID” para las transacciones.

Se observa que la mayoría de los puntos anteriores parecen ser negativos, lo que sucede es que algunas características han sido “sacrificadas” con el objetivo de poder lograr el manejo de grandes volúmenes de información, que con los sistemas relacionales ya no se puede alcanzar. Según la encuesta 2021 desarrollada por Stack OverFlow (Developer Survey 2021) MongoDB se coloca como el sistema de Base de Datos de tipo “NoSQL” mas utilizado con el 27.7% (Stack_0001, 2021).



Actividad: Instalación de MongoDB.

Nota: En el “Anexo 6” se encuentra la “Instalación de MongoDB”.



Actividad: Inserta datos en MongoDB.

Nota: Hacer un programa en Python que haga lo anterior.

```

1 import pymongo # Cargamos la librería para conectar Python con MongoDB
2 from pymongo import MongoClient # Importamos un cliente de MongoDB de la librería
3 cliente = MongoClient() # Creamos un cliente de MongoDB
4 db = cliente['agenda'] # Accesar a la base de datos: agenda
5 coleccion = db['clientes'] # Crear e insertar datos en la colección: clientes
6 registro = { "nombre": "Juan", "telefono": "4431245789"} # Datos a Insertar
7 operacion = coleccion.insert_one(registro) # Insertamos registro en nuestra colección
8 print("\n")
9 for x in coleccion.find():
10     print(x)
11 print("\n")

```

```
{'_id': ObjectId('620c6e0e841812f84c21a344'), 'nombre': 'Juan', 'telefono': '4431245789'}
```

También está surgiendo la vertiente denominada "NewSQL", donde lo que se busca es tratar de lograr el rendimiento de los sistemas NoSQL pero tratando de conseguir los conceptos "ACID" de las bases de datos tradicionales. El término lo acuñó por primera vez Matthew Aslett en el año 2011 en un intento de proponer este nuevo tipo de esquema de bases de datos..

3.3 Bases de datos distribuidas

Un sistema de bases de datos distribuidas es una colección de múltiples bases de datos relacionados de manera lógica que se localizan en los nodos de un sistema distribuido. A estos sistemas se les llama de manera genérica “DDBMS” (“*Distributed Database Management System*”) (Tamer, 2020).

3.4 Data Center

Un “*Centro de Datos*” ó “*Datacenter*” (por su nombre mas conocido en inglés) es un lugar físico donde se encuentran los recursos para el procesamiento de la información (computadoras, equipos, redes, etc.). Debido a que es mas común el término “*Datacenter*” (a veces se escribe separado “*Data Center*”), así se le denominará de hoy en adelante. Si tenemos 3 tareas, se pueden distribuir entre 3 servidores para que puedan ser procesados al mismo tiempo, lo que se convierte en una “*Computación Distribuida*” (figura 16):

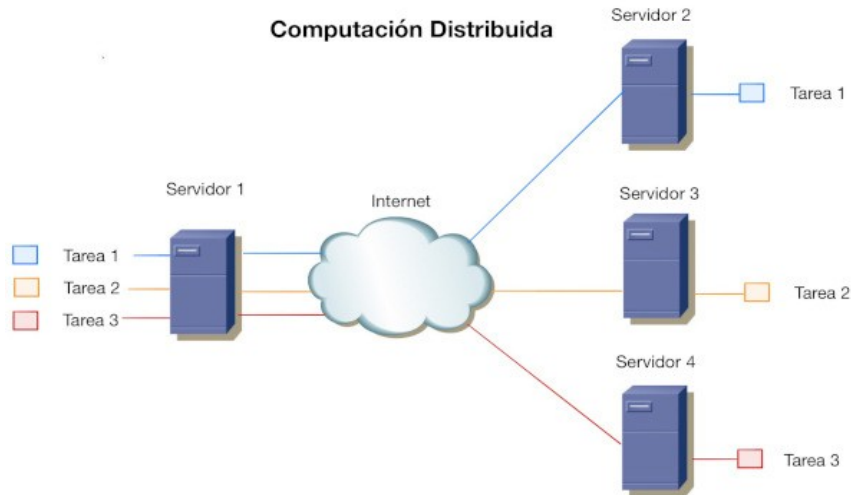


Figura 16: Computación Distribuida.

Para acelerar el procesamiento, cada una de las 3 tareas anteriores se puede a su vez dividir en pequeñas tareas que se puedan procesar en cada uno de los servidores, por lo cual se convierte en una “Computación Paralela y Distribuida” al mismo tiempo (figura 17):

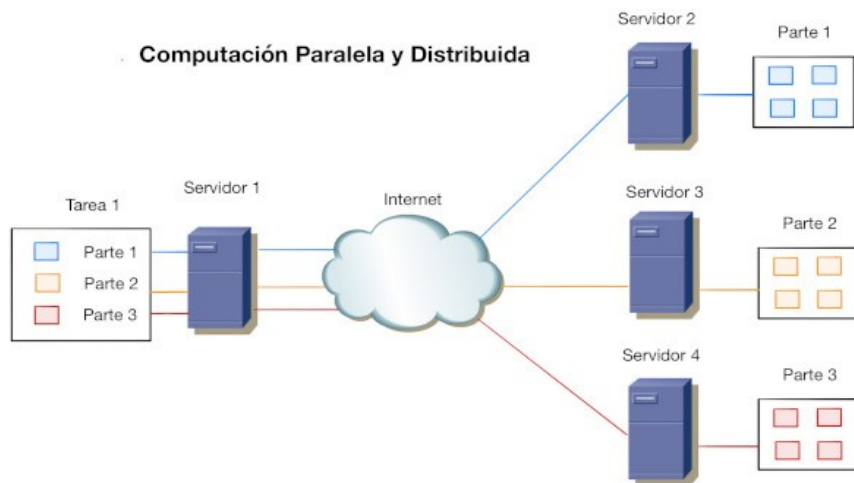


Figura 17: Computación Paralela y Distribuida



Actividad: Instalación de un Cluster

Nota: Armar un Cluster usando 2 computadoras físicas. Para ello es necesario conectarlas en red y configurarlas para que trabajen como maestro y esclavo respectivamente e instalar una versión de MPI para que puedan resolver un solo problema usando cómputo paralelo.

Nota: En el “Anexo 6” se encuentran las instrucciones completas para la “Instalación de un Cluster”, utilizando como computadoras físicas 2 Raspberry Pi.



pueda
funcionamiento.

Actividad: Imprime Hola Mundo desde un nodo de MPI.

Nota: Hacer un programa en Python que utilice el estándar MPI, y que cada noco imprima el “*Hola Mundo!*”, por lo que es necesario tener un Cluster disponible para su

```
1 from mpi4py import MPI
2 size = MPI.COMM_WORLD.Get_size()
3 rank = MPI.COMM_WORLD.Get_rank()
4 name = MPI.Get_processor_name()
5 print("\nHola Mundo con MPI")
6 print("El tamaño del Cluster es: " + str(size))
7 print("Mi rango es: " + str(rank))
8 print("Mi nombre es: " + str(name))
```

3.5 Cómputo en la nube

El incremento en la cantidad de los datos ha requerido construir dispositivos con mayor cantidad de almacenamiento y procesamiento, por lo cual, un usuario con una computadora de escritorio ya no puede manejar este tipo de información, así que el siguiente paso es llevar la información y el procesamiento a servidores que se encuentran en Internet (ó en la “*nube*” como se le denomina comúnmente).

El “*Cómputo en la Nube*” ó “*Cloud Computing*” (por su nombre en inglés), es “*es el uso de una red de servidores remotos conectados a internet para almacenar, administrar y procesar datos, servidores, bases de datos, redes y software*” (Wiki_0014, 2022).

3.5.1 Software como Servicio

Conforme la tecnología avanzaba, se logró tener dispositivos con capacidad de almacenamiento cada vez mayor y a un menor precio, por lo que el cómputo en la nube ha evolucionado de acuerdo a estos parámetros. Primero se empezaron a ofrecer servicios en la nube conocidos como “Software como Servicio” ó “*SaaS*” (“*Software as a Service*” por sus siglas en inglés), en el se cual, un usuario podría acceder a servicios que se encuentren alojados en servidores en Internet.

Uno de los ejemplos mas comunes es la suite de Google, que empezó con “*Gmail*”, el cual es un servicio de correo electrónico que fue incorporando mas características con el tiempo, como el poder abrir documentos en línea y posteriormente editarlos también, lo que finalmente se convertiría en “*Google Docs*”. Este conjunto de servicios de Google puede ser usado a través de un navegador Web y una cuenta gratuita de Google.

3.5.2 Plataforma como Servicio

Posteriormente, no sólo se ofrecieron servicios sino también ambientes completos de trabajo y desarrollo denominados “Plataforma como Servicio” ó “*PaaS*” (“*Platform as a Service*” por sus siglas en inglés), ejemplos de ellos son “*Google App Engine*” y “*Windows Azure*”.

3.5.3 Infraestructura como Servicio

La siguiente evolución fue ofrecer servidores y equipos de red completos para que puedan ser utilizados por los usuarios que puedan pagar por estos servicios, denominados “Infraestructura como Servicio” ó “IaaS” (“Infrastructure as a Service” por sus siglas en inglés), ejemplos de ellos son “AWS” y “Google Cloud”.

3.6 Virtualización

En ambientes donde se requiere tener muchas aplicaciones, cada una de ellas con requisitos de software diferente, nos lleva a tener una administración muy complicada en un servidor para poder satisfacer todas las necesidades de cada aplicación, por lo que se empezó a usar la “virtualización”.

La “virtualización” se puede definir como “la tecnología que permite ejecutar máquinas virtuales con diferentes sistemas operativos en una misma máquina física” (Raya, 2010), es decir, tener una sola computadora “física” donde se puedan ejecutar “múltiples sistemas operativos” como si tuviéramos varias computadoras.

Aunque la virtualización se empezó a usar de manera experimental desde la década de los 60’s en los Mainframes de IBM, no fue sino hasta finales de los 90’s cuando las computadoras lograron un mayor poder de cómputo gracias a los avances de la tecnología, por lo que se pudo implementar la virtualización de manera mas generalizada logrando realizar aplicaciones de muchos tipos.



Actividad: Instalación de una máquina virtual.

Nota: Para este ejemplo se usa VirtualBox para instalar una máquina virtual. En el Anexo 8 se encuentran las instrucciones para la “Instalación de una máquina virtual”.

3.6.1 Arquitecturas de Virtualización

Existen varias formas de tipos de arquitecturas de virtualización, aunque podríamos clasificarlas básicamente en 2:

- Virtualización sin un sistema operativo anfitrión.
- Virtualización a partir de un sistema operativo anfitrión.

3.6.2 Virtualización sin un sistema operativo anfitrión

En esta técnica (también denominada “*Hypervisor*”) es una arquitectura donde se instala un software justo encima del hardware, es decir, es un software que controla por completo la computadora y que se encargará de realizar la virtualización de múltiples sistemas operativos.

Este tipo de arquitectura es el mas usado en ambientes de servidores, ya que es dedicada, es decir, la computadora física sólo se puede usar para virtualizar y se aprovechan todos los recursos físicos de la computadora.

3.6.3 Virtualización a partir de un sistema operativo anfitrión

En esta segunda técnica, se tiene una computadora con algún sistema operativo instalado (denominado “*sistema nativo*”), y en él, se instala un software que se encargará de la virtualización. Este software de virtualización es el que podrá ejecutar múltiples sistemas operativos aunque tendrán

que compartir recursos en tiempo de ejecución con el sistema nativo que lo soporta. En caso de una falla en el sistema nativo, fallará también la ejecución de las máquinas virtuales.

Esta técnica, aunque no es la más eficiente para virtualizar, es la más usada entre los usuarios finales que quieren virtualizar varios sistemas, pero mantener en funcionamiento su sistema nativo.

3.7 Microservicios

Conforme ha pasado el tiempo las aplicaciones de software han crecido en complejidad, trabajando primero en computadoras pequeñas y centralizadas, para posteriormente convertirse en aplicaciones que corren en múltiples computadoras (“*cluster*”) y situados en muchos lugares (distribuidas y en la nube).

El desarrollar este tipo de aplicaciones modernas ha llevado a buscar nuevas estrategias para poder construir estas aplicaciones así como llevar a cabo su mantenimiento. Inicialmente un proyecto de software involucraba una sola aplicación que hacía todas las operaciones de manera interna (interfaz gráfica, acceso a base de datos, búsqueda e impresión de información, etc.). Para facilitar la interacción interna de estas aplicaciones se fue desarrollando una nueva forma de trabajo, en donde se hacen pequeñas aplicaciones independientes que pueden trabajar en conjunto y a la que se le denominó “*Microservicios*”. Un “*Microservicio*” es un pequeño proceso de software que trabaja de manera independiente y que puede ser actualizado sin afectar a otros procesos (Davis, 2021).

3.8 Contenedores

El uso de la virtualización ayudó a poder tener en una misma computadora física, varias computadoras virtuales, logrando la ejecución de diferentes sistemas operativos, pero cuando se tiene una gran cantidad de aplicaciones, y cada una de ellas requiere diferentes recursos para su correcto funcionamiento (bases de datos, compiladores, entornos de desarrollo, etc.), entonces ya no es tan fácil gestionar todo esto en una sola computadora.

Para solucionar lo anterior se creó el concepto de “*contenedores*”, el cual es una técnica que genere un bloque aislado para cada aplicación, con sus respectivos recursos a utilizar, lo cual permite la múltiple ejecución de aplicaciones sin que entren en conflicto unas con otras. En ciencia de datos se requiere la ejecución de múltiples aplicaciones con requerimientos diferentes, lo cual hace que los “*contenedores*” sean una una gran opción para trabajar con ellos al hacer un proyecto.

3.9 Control de Versiones

A parte de las requerimientos de cada una de las aplicaciones que se realizan, también existe el problema de las versiones que se van realizando de cada aplicación, ya que constantemente se hacen cambios a los códigos (lo que en software se le denominó “*Beta Perpetuo*”). Para llevar un mejor control de cada versión surgieron los “*Sistemas de Control de Versiones*” ó “*VCS*” (“*Version Control System*”, por sus siglas en inglés).

Los “*VCS*” rápidamente se hicieron populares y emigraron a la nube, facilitando así la interacción de varios programadores que trabajan al mismo tiempo sobre un mismo software y así evitar el problema y las posibles confusiones de las versiones de un mismo software.

Inicialmente a los “VCS” incluían pocas opciones y utilidades, pero con el tiempo se fueron agregando más. Actualmente dentro de los mismos “VCS” se encuentran almacenados no solamente el código, sino también los conjuntos de datos con los que se van a trabajar (datasets), lo cual los hace muy prácticos para trabajar dentro del área de ciencia de datos. La plataforma que mas creció fue GitHub y actualmente es la más utilizada.

3.10 Blockchain

Uno de los problemas que surgen cuando se manejan grandes cantidades de datos, especialmente cuando esos datos son personales ó confidenciales, es la seguridad. Una técnica para el manejo de información más segura es el “Blockchain”, en la cual nunca se borran las transacciones que se realizan, si no que para hacer una nueva transacción se agregan nuevos registros en la base de datos que van encadenados y cifrados a su anterior transacción del mismo usuario, por lo que el historial dentro de la base de datos va quedando cifrada y “encadenada en bloques” (de ahí su nombre de “Blockchain”).

Además del proceso anterior de encadenamiento, la base de datos con la que se trabaja está replicada en múltiples servidores, por lo que, para mantener la “atomicidad” de una transacción, ésta se tiene que hacer en todos los servidores, lo cual lo hace en general un sistema lento pero mucho más seguro.

Debido a su seguridad se ha adoptado el blockchain para varias aplicaciones, principalmente para el uso de las criptomonedas como el “Bitcoin” y mas recientemente la creación de “Tokens no Fungibles” ó “NFT” (“Non Fungible Token” por sus siglas en inglés), el cual sirve para identificar obras digitales o de cualquier otro tipo con un carácter de único y poderse ofrecer a la venta. Actualmente también ya se ha empezado a hacer ciencia de datos dentro de estos tipos de sistemas por la gran cantidad de datos que empiezan a almacenar.

Este es un esquema de una transacción con Blockchain, donde hay 2 o más usuarios conectados de manera Web al sistema, el cual hará las transacciones de cada uno de ellos y hará el encadenamiento usando los 2 servidores (figura 18):

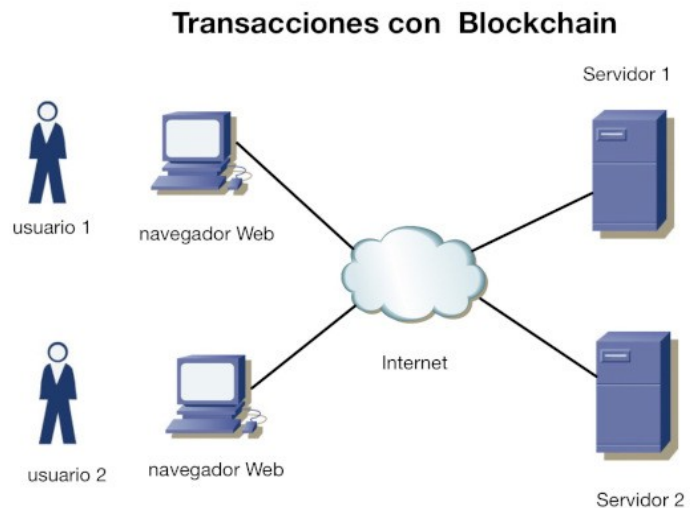


Figura 18: Transacciones con Blockchain.

Ejemplo: se tienen 2 usuarios (ana y juan) y se hacen las siguientes transacciones:

- ana abre su cuenta con \$500 pesos (génesis).
- Luego ana deposita \$100 pesos.
- juan abre su cuenta con \$300 pesos (génesis)
- ana retira \$200 pesos.
- juan deposita \$600 pesos.
- juan retira \$50 pesos.
- ana deposita \$250 pesos.
- ana retira \$150 pesos.

El encadenamiento (Blockchain) queda de la siguiente manera en las bases de datos (figura 19):

Registros en un Blockchain

No. de Operación (consecutivo)	Operación Anterior	Tipo de Operación	Monto	Cliente
1	0	0	500	ana
2	1	0	100	ana
3	0	0	300	juan
4	2	1	200	ana
5	3	0	600	juan
6	5	1	50	juan
7	4	0	250	ana
8	7	1	150	ana

Figura 19: Registros en un Blockchain.

3.11 Dataset

Según Wikipedia, estas son las características de un “Dataset” (Wiki_0006, 2021):

- Un “Dataset” es un conjunto de datos que contiene los valores para cada una de las variables, como por ejemplo la altura y el peso de un objeto, que corresponden a cada miembro del conjunto de datos.
- Cada uno de estos valores se conoce con el nombre de dato.
- El conjunto de datos puede incluir datos para uno o más miembros en función de su número de filas.
- Los Conjuntos de datos tan grandes que aplicaciones tradicionales de procesamiento de datos no los pueden tratar, a eso se le denomina “Big Data”.

Características de un buen “Dataset” (Foxwell, 2020):

- Localizable.
- Accesible.
- Interoperable.
- Reusable.

Un "Dataset" requiere un "Diccionario/Metadatos" que describa el contenido del "Dataset".

Información a tomar en cuenta al construir un archivo de Metadatos (Foxwell, 2020):

- Nombres.
- Descripciones.
- Tipos de datos.
- Unidades de medida.
- Valores de rangos permitidos.
- Codificaciones especiales.



Actividad: Lee los últimos datos del COVID en México del repositorio oficial de Johns Hopkins University.

Nota: Se requiere instalar la biblioteca "covid" para poder conectar Python con el repositorio oficial de la Universidad Johns Hopkins y obtener estadísticas sobre el Covid.

```
1 from covid import Covid # Cargamos la biblioteca para el manejo de datos de Covid
2 covid = Covid() # Creamos nuestro objeto para hacer consultas
3 lista_id_pais = covid.list_countries() # Cargamos e imprimimos la lista de "id" de cada país
4 print("\nLista de id por país:\n\n", lista_id_pais)
5 mexico_casos = covid.get_status_by_country_id(117) # Para extraer los últimos datos sobre México
6 print("\nÚltimos datos sobre México:\n\n", mexico_casos, "\n")
```

La salida de los datos para el 15 de febrero del 2022 es:

Últimos datos sobre México:

```
{'id': '117', 'country': 'Mexico', 'confirmed': 5300537, 'active': None, 'deaths': 312965, 'recovered': None, 'latitude': 23.6345, 'longitude': -102.5528, 'last_update': 1644988848000}
```

Objetivo: Conocer los conceptos básicos de Estadística y su aplicación en el mundo real actual.

“La estadística es la primera de las ciencias inexactas”

Edmond de Goncourt

Edmond Louis Antoine Huot de Goncourt (1822-1896) fue un escritor y crítico francés que escribió la obra "El Diario", que retrata y critica la sociedad parisina de finales del siglo XIX.



Para iniciar con el análisis de datos partiremos desde el área de la estadística, la cual nos permitirá conocer las características de los datos con los que estamos trabajando. Por medio de la estadística podemos ver el comportamiento y la tendencia que sufren los datos y podemos inferir los posibles valores que tendrán nuestros datos en el futuro de acuerdo a su historial. A pesar de haber muchos métodos nuevos como el Deep Learning, muchos de los problemas se pueden resolver utilizando solamente estadística, por lo cual es muy importante su conocimiento y es la base de mucho de lo que se hace actualmente.

4.1 Población y Muestra

4.1.1 Población

Población es un conjunto de elementos similares a los que se les someterá a un análisis estadístico. Este conjunto puede estar integrado por objetos como aviones, mesas, personas, estrellas, etc.. También pueden ser valores que nos representan variables físicas como temperatura, presión, humedad, etc..

4.1.2 Muestra

En muchas ocasiones la población a analizar puede ser muy grande o muy compleja y tardada de analizar, por lo cual se elige escoger una "muestra", la cual es un subconjunto de la población y que debe ser "representativa" de la población. La muestra puede ser pequeña y fácil de analizar, pero también, puede ser que no represente a la población de la cual se extrajo, por lo cual se dice que es una muestra "sesgada". El análisis de una muestra "no representativa" ó "sesgada", nos llevará a conclusiones y suposiciones erróneas.

4.2 Histogramas

En el área de estadística se utiliza mucho el histograma, el cual es un gráfico donde se representa el conjunto de datos a analizar y se representa en forma de barras, donde el tamaño de las barras es proporcional a la cantidad de datos a representar.

A continuación se muestra un histograma de los casos de contagios de COVID-19 en México a principios del año 2022 (figura 20), el cual se obtuvo del sitio oficial del Gobierno de México (Conacyt, 2022):

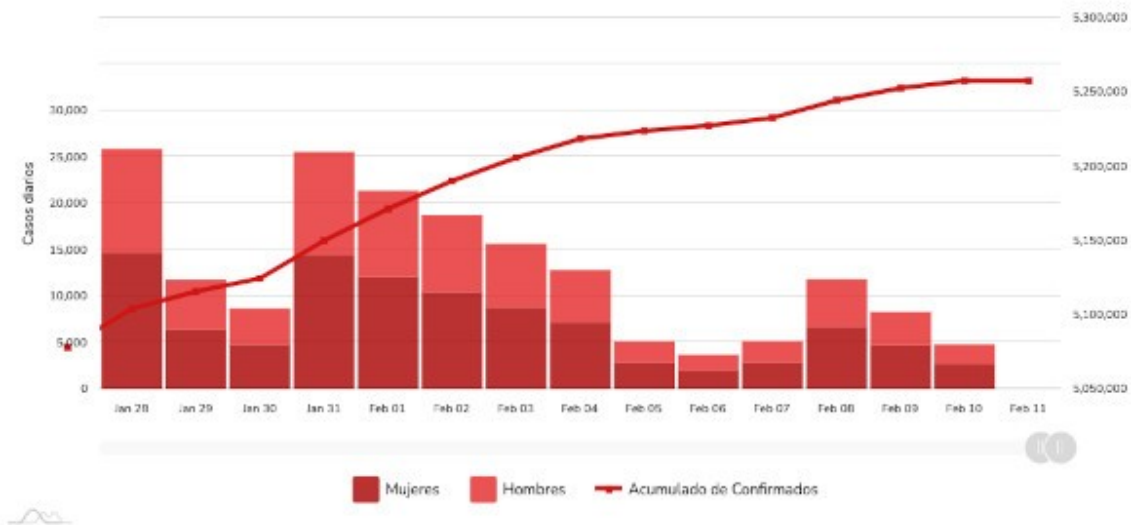


Figura 20: Histograma de contagios de COVID-19 en México

4.3 Mínimo, Máximo y Rango

4.3.1 Mínimo y Máximo

Dentro de un conjunto de datos, el "máximo" es el valor mas alto y el "mínimo" es el valor más bajo de todos. Estos puntos son importantes para delimitar el área donde se distribuyen los valores de nuestros datos. A continuación se muestra la gráfica de temperaturas máximas y mínimas que se registran para la ciudad de Morelia (figura 21), según el sitio "Weather Spark" (Spark, 2022):

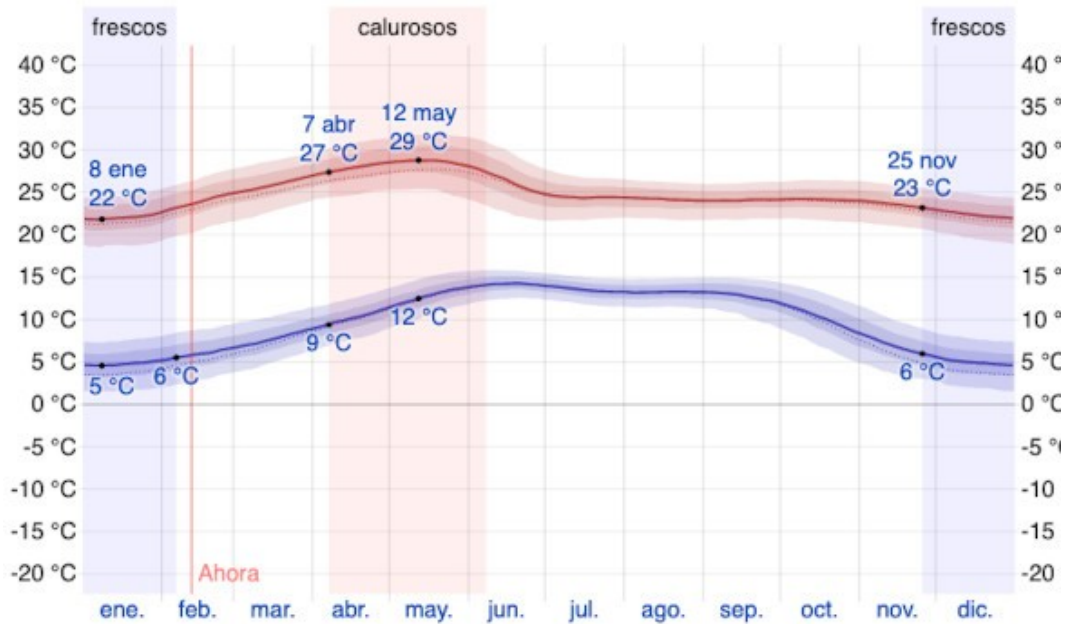


Figura 21: Temperatura anual con mínimos y máximos en la ciudad de Morelia

4.3.2 Rango

“Rango” es el conjunto de datos (intervalo) que existe entre el valor máximo y mínimo de un conjunto de datos. El rango nos ayuda a saber de manera general el grado de dispersión de nuestros datos (la varianza y la desviación estándar nos indican con exactitud la dispersión estadística).

4.4 Medidas de Tendencia Central

4.4.1 Media

La “Media Aritmética” (ó “Promedio” como se le dice comúnmente), es la suma de todos los componentes entre el número de componentes (media aritmética) (ecuación 1).

$$AM = \frac{1}{n} \sum_{i=1}^n a_i = \frac{a_1 + a_2 + \dots + a_n}{n} \quad (1)$$

4.4.2 Mediana

Dado un conjunto de datos ordenados, la “mediana” es el valor que se encuentra al centro de todos los valores. Si el número de datos es par, entonces la “mediana” es la media de los 2 valores centrales.

4.4.3 Moda

“Moda” es el valor que mas veces se repite dentro de un conjunto de datos.

4.5 Medidas de Dispersión

4.5.1 Dispersión

Como se mencionó anteriormente, la “*dispersión*” es muy importante, ya que nos indica el grado de separación de nuestros datos. Otra manera de expresarlo, es que la dispersión es, qué tanto se estiran ó se comprimen, y esto se puede observar a través de la siguiente gráfica (figura 22) desarrollada por “*JRBrown*” donde se observan 2 conjuntos de datos que tienen el mismo promedio, pero la muestra azul tiene mayor dispersión y la roja menos dispersión (JRBrown, 2022):

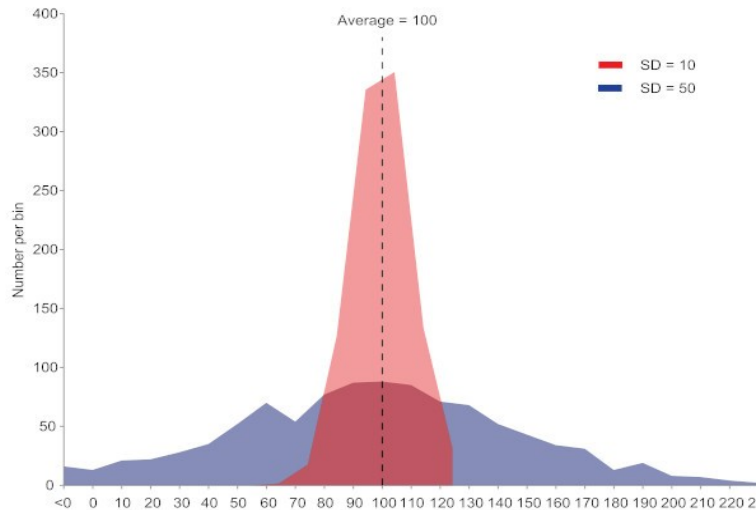


Figura 22: Cantidad de dispersión de 2 muestras

4.5.2 Varianza

La “*varianza*” (también denominada “*variancia*”) es una medida de dispersión, la cual nos sirve para representar la variabilidad de un conjunto de datos con respecto a la media aritmética de los mismos datos.

La unidad de medida de la varianza es la misma unidad de medida de los datos que se están analizando, por ejemplos si se están analizando medidas en metros, entonces la varianza también estará dada en metros. La varianza siempre tiene valores iguales o mayores a 0.

4.5.3 Desviación Estándar

La “*desviación estándar*” (también conocida como “*desviación típica*”) es una medida para cuantificar la dispersión de un conjunto de datos y se calcula como la raíz cuadrada de la varianza.

Si la desviación estándar es baja, entonces indica que los datos están agrupados cerca de la media del conjunto de datos. Y al contrario, entre más alta sea la desviación estándar, indica que los valores extienden sobre un rango mas amplio de datos.

M. W. Toews dibujó una curva de distribución normal y colocó rangos de valores que se va obteniendo para la desviación estándar (Toews, 2022), la cual se puede observar a continuación (figura 23):

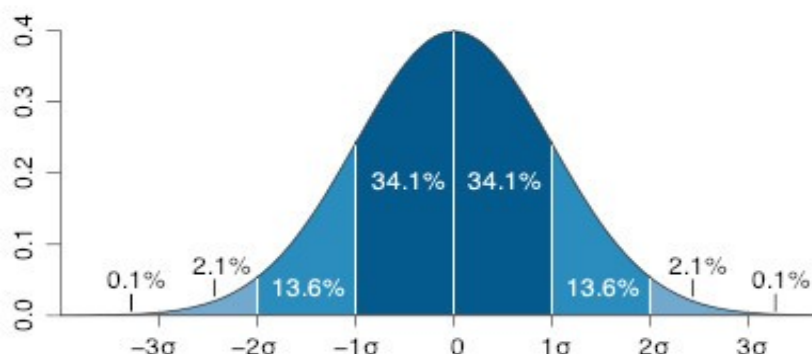


Figura 23: Valores de la desviación estándar para una distribución normal

4.6 Correlación

La “*correlación*” es un valor que nos indica la cantidad de proporcionalidad existente entre 2 variables, cuando una de ellas varía sistemáticamente con respecto a los valores homónimos de la otra.

Si la correlación es alta entre las variables A y B, esto nos indica que si los valores de A aumentan o disminuyen, también lo hacen los valores de B. Sin embargo, la correlación entre 2 variables no implica directamente que haya alguna relación de causalidad entra ambas.

Por ejemplo (figura), si tenemos una tabla con datos de Edad/Presión Sistólica en hombres, se observará que conforme aumenta la “*Edad*” (variable A) también va aumentando la “*Presión Sistólica*” (variable B).

Edad	Presión Sistólica
16	105
19	110
25	115
30	120
40	125
50	130
60	135

Figura 24: Tabla de Edades/Presión Sistólica.

4.7 Regresión lineal

La “*regresión lineal*” es un modelo matemático para predecir una respuesta (variable dependiente) de acuerdo a una función de otra variable (variables independiente), es decir, dado un conjunto de datos podemos crear un modelo que nos permita predecir los valores futuros a que pueden llegar los datos.



Actividad: Usando regresión lineal calcular el peso de una persona.

Nota: Se requiere instalar Matplotlib para hacer la gráfica.

Datos: Juanito es un niño de 10 años y su doctor particular lleva el siguiente registro de su peso:

- Al año pesaba 6 kilogramos.
- A los 5 años pesaba 15 kilogramos.
- A los 10 años pesaba 40 kilogramos.

Usando regresión lineal calcule ¿cuánto pesará Juanito cuando tenga 20 años?

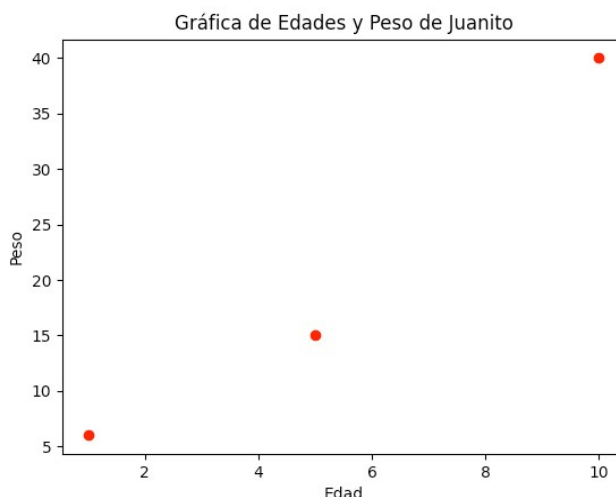
Solución: Tenemos la edad, que será nuestra variable independiente (x). El peso depende de la edad, y se convierte en la variable dependiente (y). De lo anterior nos queda la siguiente tabla:

x	1	5	10
y	6	15	40

Calculamos su “*diagrama de dispersión*”, cargando los datos a una lista y los graficamos usando Matplotlib:

```
1 # Importando matplotlib para graficar
2 import matplotlib.pyplot as plt
3 # Creamos una lista para indicar las edades
4 x_lista = [1, 5, 10]
5 # Creamos una lista para indicar los pesos
6 y_lista = [6, 15, 40]
7 # Hacemos un ciclo para imprimir todos los puntos
8 contador = 0
9 for ciclo in x_lista:
10     plt.plot(ciclo, y_lista[contador], marker="o", color="red")
11     contador = contador + 1
12 # Colocamos un título a nuestra gráfica y a nuestros ejes
13 plt.title("Gráfica de Edades y Peso de Juanito")
14 plt.xlabel("Edad")
15 plt.ylabel("Peso")
16 # Mostramos la gráfica en pantalla
17 plt.show()
```

La salida en pantalla es la siguiente:



En los datos se observa una tendencia lineal. Para modelar el comportamiento de los datos se propone una recta, donde conocemos “x”, “y”, y buscaremos el valor de “m” y “b” para poder tener nuestra función lineal que represente a nuestro modelo (figura):

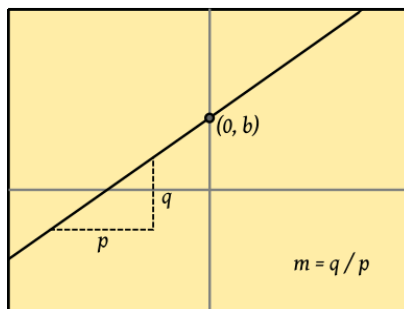


Figura 25: Gráfica de una Función Lineal

La ecuación de la gráfica anterior es (ecuación 2)

$$y = mx + b \tag{2}$$

A partir del análisis de la ecuación de la recta, se obtienen las siguientes ecuaciones normalizadas:

$$\sum y = m \sum x + nb \tag{3}$$

$$\sum xy = m \sum x^2 + b \sum x \tag{4}$$

Se requiere calcular los siguientes datos:

$$\sum x \tag{5}$$

$$\sum y \tag{6}$$

$$\sum x^2 \tag{7}$$

$$\sum xy \tag{8}$$

Se hacen los siguientes cálculos:

$$\sum x = x_0 + x_1 + \dots + x_n = 1 + 5 + 10 = 16 \tag{9}$$

$$\sum y = y_0 + y_1 + \dots + y_n = 6 + 15 + 40 = 61 \tag{10}$$

Después:

$$\sum x^2 = x_0^2 + x_1^2 + \dots + x_n^2 = (1)^2 + (5)^2 + (10)^2 = 1 + 25 + 100 = 126 \tag{11}$$

$$\sum xy = x_0 y_0 + x_1 y_1 + \dots + x_n y_n = (1)(6) + (5)(15) + (10)(40) = 6 + 75 + 400 = 481 \quad (12)$$

Se hacen los siguientes cálculos:

$$\sum y = m \sum x + nb \quad (13)$$

$$61 = 16m + 3b \quad (14)$$

$$\sum xy = n \sum x^2 + b \sum x \quad (15)$$

$$481 = 126m + 16b \quad (16)$$

Despejando "m" en la ecuación 14 tenemos:

$$m = \frac{61 - 3b}{16} \quad (17)$$

Despejando "m" en la ecuación 16 tenemos:

$$m = \frac{481 - 16b}{126} \quad (18)$$

Igualando:

$$\frac{61 - 3b}{16} = \frac{481 - 16b}{126} \quad (19)$$

Nos da lo siguiente:

$$3.8125 - 0.1875b = 3.817460317460317 - 0.126984126984127b \quad (20)$$

Despejando b:

$$-0.060515873015873b = 0.004960317460317 \quad (21)$$

$$b = -0.081967213114747 \quad (22)$$

Sustituyendo b en la ecuación 14:

$$16m + 3b = 61 \quad (23)$$

$$16m = 61 - 3b \quad (24)$$

$$m = \frac{61 - 3b}{16} \quad (25)$$

$$m = 3.8125 - 0.0153688524599015 \quad (26)$$

$$m = 3.797131147540985 \quad (27)$$

Finalmente encontramos nuestra ecuación de la recta, sustituyendo “m” y “b” en la ecuación de la recta:

$$y = 3.79x - 0.081 \quad (28)$$

Para predecir: ¿Cuánto pesará Juanito cuando tenga 20 años?

Sustituimos $x = 20$ en nuestra ecuación de la recta:

$$y = (3.79)(20) - 0.081 = 75.8 - 0.081 = 75.719 \text{ kilogramos} \quad (29)$$

Capítulo

Probabilidad

5

Objetivo: Conocer los conceptos básicos de Probabilidad y su aplicación en el mundo real actual.

“El destino se ríe de las probabilidades”

Edward Bulwer-Lytton

Edward George Earle Bulwer-Lytton (1803-1873) fue un periodista británico que destacó en la poesía, la novela y la política donde llegaría a ser parte del parlamento inglés y posteriormente Secretario de Estado para las Colonias. Tuvo una prolífica obra literaria en gran variedad de géneros como ficción histórica, misterio, novela romántica, ocultismo y ciencia-ficción. Se le recuerda por la frase "La pluma es más fuerte que la espada".



Otra área importante para el manejo y análisis de datos es la “*Probabilidad*”, ya que el solo hecho de llevar una estadística sobre ciertos hechos nos lleva a calcular de manera inmediata un conjunto de valores probables para cada hecho analizado. Los datos que provienen de eventos aleatorios que se suscitan en muchas áreas nos lleva a cálculo de las posibilidades de que ciertos eventos ocurran en un cierto tiempo. En este punto radica la importancia de estudiar este tema tan importante que se verá en este capítulo y que en conjunto con la estadística son la base del análisis de la “*Ciencia de Datos*”.

5.1 Conceptos

5.1.1 Definición de Probabilidad

La “*Teoría de la Probabilidad*” es una rama de la matemática que se encarga de estudiar los fenómenos aleatorios y es una medida para determinar si un evento ocurrirá o no. Por ejemplo, utilizando la probabilidad podemos responder a la pregunta ¿hoy va a llover?



Existen diferentes definiciones ó interpretaciones de “*probabilidad*”, como son la clásica, la lógica, la subjetiva, la de frecuencia y la de propensión, que finalmente pueden agruparse en 2 grandes áreas de la probabilidad y la estadística (Sucar, 2015):

- Objetiva (clásica, frecuencia y propensión).
- Epistemológica (lógica, subjetiva).

Del trabajo "*Probability Theory: The Logic of Science*", realizado por Edwin Thompson Jaynes (Jaynes, 2003), se establecen algunos conceptos básicos de la probabilidad que son los siguientes:

- Es representada por números reales.
- Correspondencia cualitativa con sentido común.
- Consistencia.

5.1.2 Propiedades básicas

La probabilidad se expresa como un número que está entre 0 y 1:

$$0 \leq P(A) \leq 1 \quad (30)$$

Si tenemos la certeza de que el evento B ocurrirá, entonces:

$$P(B) = 1 \quad (31)$$

Si A y B son eventos mutuamente excluyentes:

$$P(A \cup B) = P(A) + P(B) \quad (32)$$

5.1.3 Espacio de Muestreo

Es el conjunto de todos los posibles resultados de un experimento aleatorio. En el caso de una moneda (en México), sería águila ó sol.



$$S = \{ \text{Águila}, \text{Sol} \} \quad (33)$$



Actividad: Lanzar 10 veces una moneda e imprimir el resultado

Nota: Hacer un programa en Python que simule lo anterior.

```

1 # Biblioteca para números aleatorios enteros
2 from random import randint
3 # Se definen valores para la moneda
4 #     aguila = 1     sol = 0
5 # Se inicializa la moneda
6 aguila = 0
7 sol = 0
8 print("\n")
9 for lanzamiento in range(1, 11):
10     valor = randint(0, 1)
11     if (valor==1):
12         moneda = "aguila"
13         aguila = aguila + 1
14     else:
15         moneda = "sol"
16         sol = sol + 1
17     print("Lanzamiento ", lanzamiento, " valor ", moneda)
18 print("\nResultado Final: Aguilas=",aguila," Soles=",sol,"\n")

```

```

Lanzamiento 1 valor sol
Lanzamiento 2 valor aguila
Lanzamiento 3 valor aguila
Lanzamiento 4 valor aguila
Lanzamiento 5 valor sol
Lanzamiento 6 valor aguila
Lanzamiento 7 valor sol
Lanzamiento 8 valor sol
Lanzamiento 9 valor aguila
Lanzamiento 10 valor aguila

Resultado Final: Aguilas= 6 Soles= 4

```

5.1.4 Ley de los grandes números

Si lanzamos una moneda muchas veces, la probabilidad de que caiga águila ó sol, tenderá a ser de 0.5 ($\frac{1}{2}$). A esto se le conoce como la “Ley de los grandes números”.



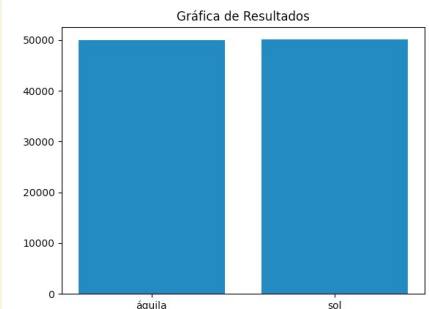
Actividad: Lanzar 100,00 veces una moneda e imprimir el resultado

Nota: Hacer un programa en Python que simule lo anterior.

```

1 from random import randint # Biblioteca para números aleatorios enteros
2 import matplotlib.pyplot as plt # Biblioteca para graficar
3 aguila = 0 # Se inicializan los contadores
4 sol = 0
5 for lanzamiento in range(1, 100001):
6     valor = randint(0, 1) # valores para la moneda: aguila=1, sol=0
7     if (valor==1):
8         aguila = aguila + 1
9     else:
10        sol = sol + 1
11 print("\nResultado Final: Aguilas=",aguila," Soles=",sol,"\n")
12 etiquetas = ["águila", "sol"] # Lista con las etiquetas
13 resultados = [aguila, sol] # Lista con los resultados
14 plt.bar(etiquetas, resultados) # Creamos una gráfica de barras
15 plt.title("Gráfica de Resultados") # Agregamos un título a la gráfica
16 plt.show() # Mostrar la gráfica en pantalla

```



Si en el programa anterior, el ciclo es pequeño (de 1 a 10) se observarán posibles diferencias significativas entre “águila” y “sol”, pero si el ciclo se incrementa a 100,000 se observará casi un resultado de 50% y 50% entre “águila” y “sol”, comprobándose la “Ley de los Grandes Números”.

5.2 Espacio de Probabilidad

5.2.1 Definición

Para entender mejor y poder generar modelos matemáticos acerca de un experimento aleatorio, en 1933 Andréi Kolmogorov introdujo el concepto de “Espacio de Probabilidad” (Kolmogorov, 1933). Un espacio de Probabilidad está constituido por 3 partes (Stanley, 2021):

- Espacio de Muestreo.
- Espacio del Evento.
- Ley de Probabilidad.

La primera parte de un “*Espacio de Probabilidad*” es el “**Espacio de Muestreo**”, el cual es el conjunto de todas las posibles salidas de un experimento (se definió en la sección 5.1.3) y está representada por el símbolo griego “ Ω ”. El espacio de muestreo puede ser de 2 maneras:

- Discreto.
- Continuo.

Algunos ejemplos de un espacio de muestreo discreto son:

- Lanzar una moneda: $\Omega = \{ \text{águila, sol} \}$
- Lanzar un dado: $\Omega = \{ 1, 2, 3, 4, 5, 6 \}$
- Números enteros par: $\Omega = \{ 2, 4, 6, 8, \dots \}$

Algunos ejemplos de un espacio de muestreo continuo son:

- Ángulo de fase de un voltaje: $\Omega = \{ \theta \mid 0 \leq \theta \leq 2\pi \}$
- Una hora: $\Omega = \{ t \mid 0 \leq t \leq 60 \text{ minutos} \}$

La segunda parte de un “*Espacio de Probabilidad*” es el “**Espacio del Evento**”, el cual es toda la colección de eventos posibles y que está representado por el símbolo “ \mathcal{F} ” y es un subconjunto de “ Ω ”.

Un ejemplo es:

- Experimento: Lanzar un dado.
- Espacio de muestreo: $\Omega = \{ 1, 2, 3, 4, 5, 6 \}$
- 2 Eventos
 - Evento 1 = { Números par } = { 2, 4, 6 }
 - Evento 2 = { Mayores de 4 } = { 5, 6 }

La tercera parte de un “*Espacio de Probabilidad*” es el “**Ley de Probabilidad**”, la cual es la probabilidad de cada uno de los eventos. Si calculamos las probabilidades para los eventos anteriores nos queda:

- $P(\text{Evento 1}) = 3 / 6$
- $P(\text{Evento 2}) = 2 / 6$

5.3 Probabilidad condicional

5.3.1 Definición

Dentro del área de ciencia de datos, es muy importante el descubrir las relaciones entre 2 eventos (por ejemplo A y B), y ver si es posible que un evento pueda causar o alterar al otro (que A cause a B), lo que indica una relación entre los eventos, lo que denominamos “*Probabilidad Condicional*”. Esta es muy importante, ya que esta relación puede desencadenar otros eventos, por ejemplo el evento B cause a C, por lo tanto podemos encontrar que si A implica a B, y B implica a C, podemos encontrar una relación entre A y C. Esto en un ejemplo real nos puede llevar a encontrar un evento que desencadene a muchos más.

5.3.2 Probabilidad Condicional

Matemáticamente la “*Probabilidad Condicional*” la podemos expresar como:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (34)$$

5.3.3 Independencia

Cuando tenemos 2 eventos, en donde ninguno de los 2 depende del otro, entonces decimos que son “*independientes*”, y matemáticamente lo podemos expresar como:

$$P[A \cap B] = P[A]P[B] \quad (35)$$

5.4 Variables Aleatorias

5.4.1 Definición

Las variables aleatorias de un evento, como lanzar un dado, se pueden mapear eventos a números, para posteriormente convertirse en parte de un dataset el cual pueda ser analizado en un proceso de ciencia de datos. Por ejemplo, se tienen 4 cartas (♠ ♣ ♥ ♦), para las cuales les construimos su espacio de probabilidad y nos queda como sigue:

- Cartas: $\Omega = \{ \spadesuit, \clubsuit, \heartsuit, \diamondsuit \}$

La probabilidad de sacar cada de una de las cartas queda de la siguiente manera:

- $P[\{ \spadesuit \}] = \frac{1}{4}$
- $P[\{ \clubsuit \}] = \frac{1}{4}$
- $P[\{ \heartsuit \}] = \frac{1}{4}$
- $P[\{ \diamondsuit \}] = \frac{1}{4}$

Ahora, para no usar los símbolos y convertirlos a números ocupamos una función X , la cual se denominará “*variable aleatoria*”, la cual de manera matemática se puede expresar como:

- $X: \Omega \rightarrow \mathbb{R}$

Por lo tanto, para no manejar símbolos y pasarlos a números, aplicamos nuestra función X a cada uno de nuestros símbolos para generar un número con el que podamos identificar a cada de los elementos, que en este caso son cartas y queda de la siguiente manera:

- $X(\spadesuit) = 1, X(\clubsuit) = 2, X(\heartsuit) = 3, X(\diamondsuit) = 4$

5.4.2 Variables Aleatorias Discretas

Son aquellas que se pueden definir y contar, como por ejemplo al lanzar una moneda (águila ó sol), lanzar un dado (del 1 al 6), etc..

5.4.3 Variables Aleatorias Continuas

Son aquellas que adquieren valores dentro de un rango limitado, por ejemplo, la temperatura del agua, el tiempo de vuelo de un avión, etc..

5.5 Distribuciones de Probabilidad

5.5.1 Definición

Con el objetivo de simular fenómenos y acontecimientos de la vida real, los científicos han desarrollado diferentes modelos probabilísticos para simular el resultado de valores que se pueden obtener de un evento, de acuerdo a una probabilidad asociada al evento. A estos modelos se les denominó "*Distribución de Probabilidad*". De acuerdo a las variables con las que se va a trabajar, las distribuciones se pueden dividir en 2 grupos:

- Si se utilizan variables discretas (distribución Binomial y de Poisson).
- Si se utilizan variables continuas (distribución Normal).

5.5.2 Distribución Binomial

Esta distribución fue propuesta por Jakob Bernoulli (figura 26) y se utilizan en modelos y simulaciones donde la respuesta de los eventos sea de tipo binaria (Si ó No, Éxito ó Fracaso, Verdadero ó Falso, etc.). Esta distribución se ajusta a muchas situaciones de la vida real como son:

- Si una persona tiene COVID-19.
- Si se hizo una tarea ó proceso.
- Si hay agua en un tanque.

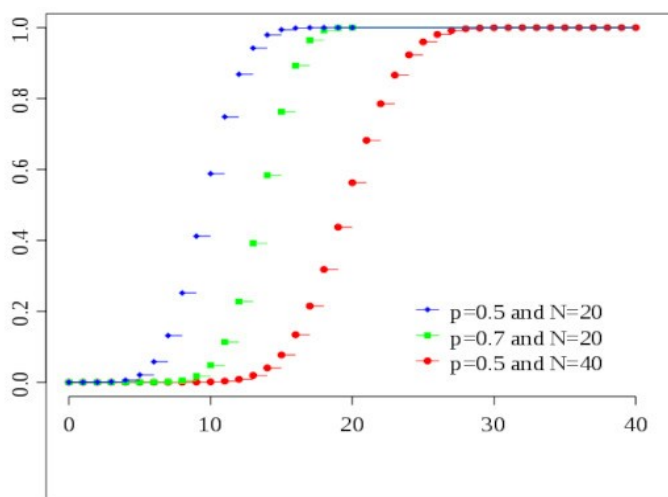


Figura 26: Distribución Binomial

5.5.3 Distribución de Poisson

Esta distribución fue propuesta por el matemático Francés Siméon Denis Poisson y se utiliza principalmente para modelar la cantidad de veces que sucede un número de eventos en un tiempo determinado (figura 27). Se puede utilizar en situaciones como:

- Número de vehículos que transitan por una calle en una hora.
- Cantidad de mensajes enviados en un día.
- Número de productos por hora que no superaron el control de calidad.

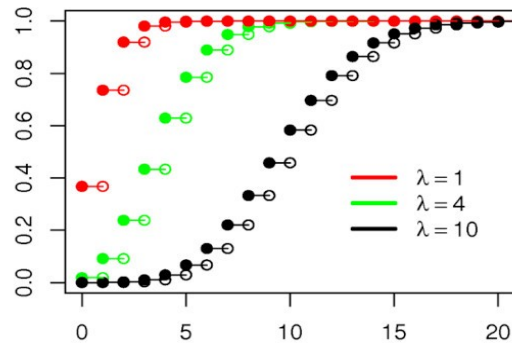


Figura 27: Distribución de Poisson

5.5.4 Distribución Normal

La "Distribución Normal" (también llamada de "Gauss" ó "Gaussiana") es una distribución de variable continua que más se utiliza en Probabilidad y Estadística y fue propuesta por el alemán Carl Friedrich Gauss (figura 28). Esta distribución es el gráfico de una función gaussiana y que tiene una forma acampanada (de ahí su nombre de "Campana de Gauss") y es simétrica. Esta distribución es la mas utilizada ya que se encontró que con ella se puede modelar muchos fenómenos físicos, biológicos, económicos, sociales y hasta psicológicos.

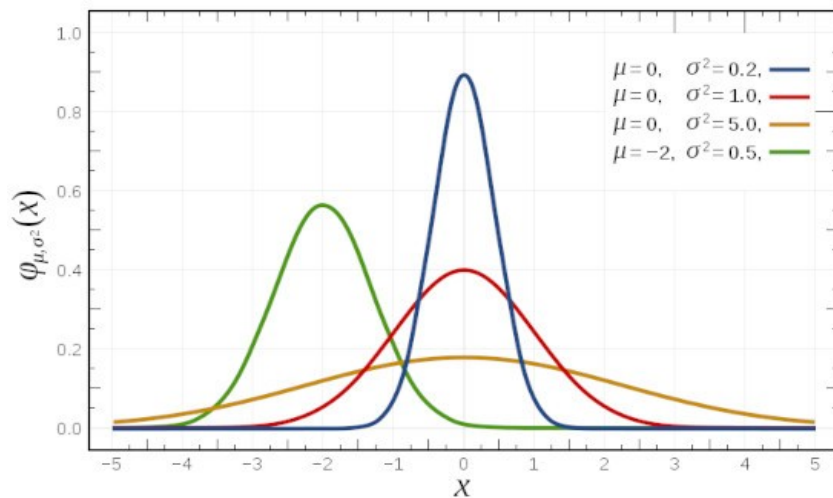


Figura 28: Distribución Normal

5.5.5 Teorema del Límite Central

Es una teoría que nos indica que cuando se tiene una muestra aleatoria de la población lo suficientemente grande, entonces la distribución de las medias muestras seguirá una distribución Normal.

Este teorema es muy útil cuando se tienen muestras aleatorias pequeñas y por lo tanto podemos recurrir a él para inferir que tendrá una distribución muy próxima a la normal. Podemos recurrir a este teorema cuando por algún factor no se puede recolectar los datos de toda una población. El que una distribución se parezca mucho a la normal es muy útil ya que nos permite utilizar varias técnicas que se pueden aplicar a este tipo de distribución.

5.6 Modelos Gráficos Probabilísticos

5.6.1 Definición

Los Modelos Gráficos Probabilísticos ó PGM (*"Probabilistic Graphical Model"* por sus siglas en inglés) es un modelo construido a partir de un grafo donde se expresa la dependencia condicional entre sus nodos utilizando variables aleatorias (figura 29). Estos modelos son ampliamente utilizados en probabilidad, estadística, aprendizaje máquina (Machine Learning) y por lo tanto en también en ciencia de datos.

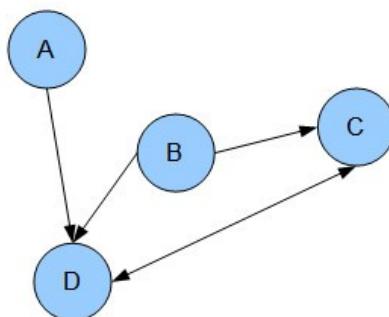


Figura 29: Modelo Gráfico Probabilístico.

5.6.2 Aplicaciones de los PGM

Algunas de las aplicaciones que se pueden hacer utilizando los PGM son:

- ¿Cuál es la probabilidad de que llueva el día de hoy?
- ¿Cuál es la probabilidad de contraer COVID dado que vivo en Morelia?
- ¿Cual es la probabilidad de que México gane el mundial de Qatar?

5.6.3 Clasificación de los PGM

De acuerdo a ciertas características podemos agrupar los PGM en 3 grandes grupos (Sucar, 2022):

- Dirigido y No-dirigidos.
- Estáticos y Dinámicos.
- Probabilística y decisiones.

5.6.4 Grafos No Dirigidos y Grafos Dirigidos

Un grafo es una forma fácil de representar las relaciones entre un conjunto de objetos. Los objetos son representados por círculos y las relaciones son las líneas o flechas. Un *"grafo no dirigido"* (figura 30) representa una una relación binaria simétrica mientras que un *"grafo dirigido"* (figura 31) es una representación binaria no simétrica:

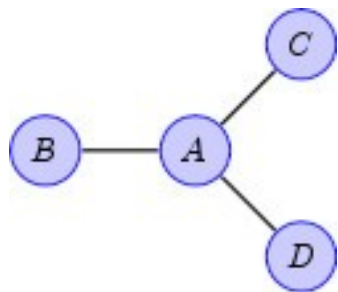


Figura 30: Grafo no dirigido.

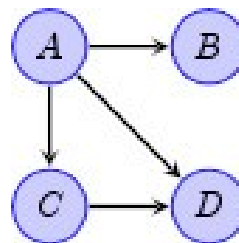


Figura 31: Grafo dirigido.

5.7 Redes Bayesianas

5.7.1 Definición

Es un PGM de tipo estático representado por un grafo cíclico dirigido que nos representa un conjunto de variables aleatorias y sus dependencias condicionales. Estas relaciones probabilísticas pueden ser usadas para muchas cosas, por ejemplo, se puede calcular ¿cuál es probabilidad de que la hierba esté seca si está lloviendo? (figura 32):

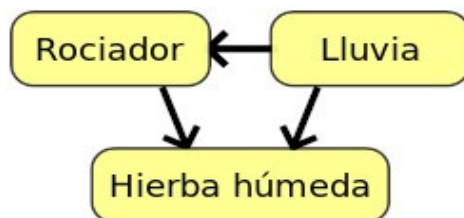


Figura 32: Red Bayesiana simple.

Para resolver la pregunta anterior (Wikipedia_0015, 2022), primero vamos a colocar las probabilidades a priori de cada nodo para pasar al otro nodo y nos queda de la siguiente manera (figura 33):

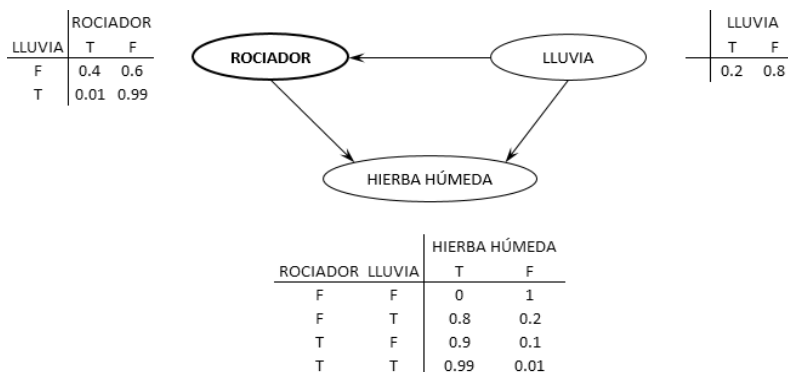


Figura 33: Probabilidades a priori de cada nodo.

Para resolver esto aplicamos el “Teorema de Bayes” y nos queda la siguiente fórmula:

$$P(G, S, R) = P(G|S, R)P(S|R)P(R) \tag{36}$$

Donde G = Hierba Húmeda, S = Rociador y R = Lluvia. Sustituyendo nos queda:

$$\left(\frac{(0.99 \times 0.01 \times 0.2 = 0.00198_{TF}) + (0.8 \times 0.99 \times 0.2 = 0.1584_{TF})}{0.00198_{TF} + 0.288_{TF} + 0.1584_{TF} + 0_{TF}} \right) \approx 35.77\% \tag{37}$$

5.8 Modelos de Markov

5.8.1 Definición

Es un modelo donde se establece una fuerte dependencia entre un evento y su evento inmediato anterior, por lo tanto, para que algo suceda en un momento t+1, solo se depende del momento anterior t. Fue propuesto por el matemático ruso Andréi Márkov en 1906 y actualmente se ocupa en una gran cantidad de aplicaciones, como predecir patrones de conducta, estados meteorológicos, predicción en negocios y finanzas, etc. (figura 31).

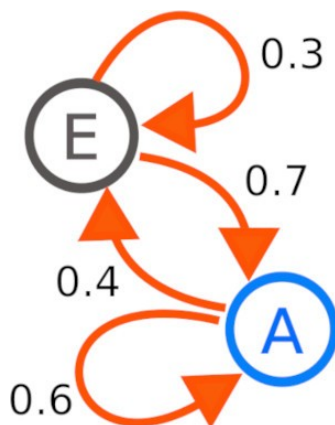


Figura 34: Cadena de Márkov

Capítulo

Machine Learning

6

Objetivo: Conocer la evolución y aplicación de los diferentes tipos actuales de Machine Learning.

“Sólo cuando las máquinas realmente piensen, podrán considerarse inteligentes”

John McCarthy

John McCarthy (1927-2011) fue Profesor Investigador de la Universidad de Stanford, el cual introdujo el concepto de "Inteligencia Artificial" en 1956 durante la Conferencia de Dartmouth, lugar donde se reunió con prestigiosos investigadores de aquella época para definir y tratar de debatir sobre el surgimiento de la Inteligencia Artificial.



Este capítulo nos lleva a los inicios del Machine Learning, las primeras ideas y conceptos que se fueron trabajando en la segunda mitad del siglo XX, así como de los primeros algoritmos y su aplicación en el mundo real.

6.1 Conceptos

6.1.1 Definición

Aunque este capítulo parecería un tema mas de Inteligencia Artificial, lo cierto es que actualmente varios algoritmos y técnicas usadas en Ciencia de Datos utilizan Inteligencia Artificial, especialmente de Machine Learning y Deep Learning, los cuales han dado un gran avance al área de Ciencia de Datos. Por lo anterior, este capítulo y el siguiente, se centrarán en Machine Learning y Deep Learning y su aplicación en Ciencia de Datos. Otro punto importante es que es difícil hablar de Ciencia de Datos sin Inteligencia Artificial y viceversa, ya que la gran cantidad de datos que se generan actualmente ha propiciado un gran impulso a la Inteligencia Artificial que bajo ciertos esquemas específicos, funcionan mejor cuando se tiene una gran cantidad de datos. También el incremento del poder de cómputo que se ha logrado actualmente ha sido importante para el manejo de grandes cantidad de información en tiempos razonables.

Para iniciar con este capítulo, se empezará con el surgimiento del concepto de “*Inteligencia Artificial*” y la creación de las primeras “*máquinas inteligentes*”. Hasta el día de hoy no se tiene alguna definición exacta de lo que es una “*máquina inteligente*” (lo que actualmente conocemos como “*Inteligencia Artificial*”) ya que es muy difícil definir lo que es “*inteligencia*”. Es por ello que el “*Aprendizaje Máquina*” ó “*Aprendizaje Automático*” (“*Machine Learning*” por su nombre en inglés, el cual es mas conocido y es el que se va a utilizar de hoy en adelante durante este libro) también es difícil de definir.

Una de las primeras definiciones (y de las más aceptadas) acerca del Machine Learning, la propuso en 1959 el científico norteamericano (y pionera de la investigación sobre Inteligencia Artificial) Arthur Lee Samuel, el cual, fue el primero que acuñó la frase “*Machine Learning*”, la que definió como “*la habilidad de aprender de una computadora sin haber sido programada de manera explícita*” (IEEE CS 1, 2022).

Otra definición la propuso el investigador norteamericano Tom M. Mitchell donde dice que el “*Machine Learning es el estudio de los algoritmos que le permitan a una computadora de manera automática mejorar a través de la experiencia*”. Esta definición apareció en su libro “*Machine Learning*” de la Editorial McGraw Hill en 1997 (Mitchell, 1997) y es uno de los mas citados.

Posteriormente al Machine Learning surgió el “*Deep Learning*” como un área del Machine Learning que utiliza como componente principal las “*Redes Neuronales*” pero con mayor cantidad de capas intermedias (este tema se tratará a profundidad en el siguiente capítulo de este libro).

Actualmente algunos autores definen al Deep Learning como una parte del Machine Learning, el cual a su vez es parte del área de Inteligencia Artificial (figura 35), aunque algunos no están totalmente de acuerdo (figura 36) y consideran que no todo el Machine Learning ni todo el Deep Learning son parte del área de Inteligencia Artificial (Wiki_0016, 2022).

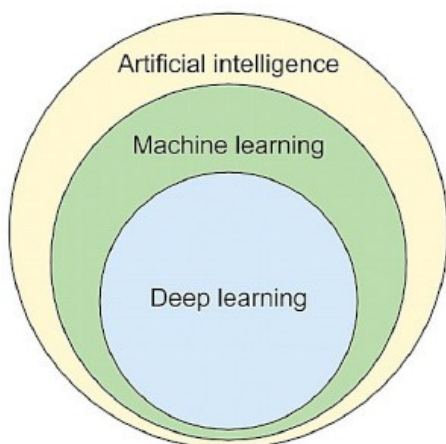


Figura 35: Subcampos de la IA

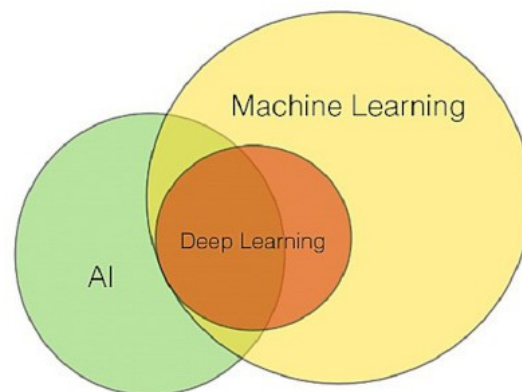


Figura 36: Algunas áreas que pertenecen a la IA

6.2 Historia

6.2.1 Primeros desarrollos

Para llegar al Machine Learning y posteriormente al Deep Learning y sus aplicaciones en Ciencia de Datos, se tuvo una larga sucesión de eventos, investigaciones y propuestas, por lo que a

continuación sólo se mencionarán algunos puntos relevantes de todo lo que se considera la historia del Machine Learning.

El primer paso para construir una máquina que pudiera procesar información la dió Alan Turing (figura 37) con su "*Máquina Automática*" en 1936, la cual es un modelo de una máquina que procesaba instrucciones que se encontraban en una cinta, la cual podría ser construida y modelada como una máquina. Posteriormente se le denominó "*Máquina de Turing*" en su honor.

Basándose en las neuronas del cerebro, los norteamericanos Warren McCulloch y Walter Pitts (figura 38) proponen el primer modelo para crear una "*neurona artificial*" en 1943. Este modelo tiene entradas y salidas binarias, las cuales se suman sus pesos ponderados a la salida, la cual se activa por medio de una señal, en este caso una función "*sigmoide*".

En 1950 Alan Turing lanza la pregunta "*¿una máquina puede pensar?*", y a partir de esta idea establece una prueba para analizar si una computadora puede llamarse "*inteligente*", la cual se conoce como la "*Prueba de Turing*".

En 1952 Arthur Samuel (figura 39) escribe el primer programa para jugar ajedrez, en una máquina IBM mientras trabajaba ahí. Este programa tenía la característica de mejorar su calidad de juego por cada vez que se utilizaba.



Figura 37: Alan Turing



Figura 38: Walter Pitts



Figura 39: Arthur Samuel

En 1956 Jhon McCarthy (figura 40), que acuñó el término "*Inteligencia Artificial*", reunió en Dartmouth College a un conjunto de investigadores y científicos por 6 semanas durante el verano (principalmente matemáticos) para debatir si una máquina puede pensar. Este evento se considera el nacimiento del área de la "*Inteligencia Artificial*".

En 1957 Frank Rosenblatt (figura 41) propone la primera red neuronal computacional la cual denominó "*Perceptrón*". Esta red fue construida para reconocimiento de imágenes y funcionaba como un clasificador. Fue codificado en una computadora IBM 704 que tenía 400 fotoceldas, las cuales eran los datos de entrada que se conectaban a la red neuronal, y como salida un identificador. En 1964 Joseph Weizenbaum (figura 42), un investigador Germano-Americano del MIT crea "*ELIZA*", un programa computacional que simula una conversación entre una persona real y una máquina. Aunque era una herramienta muy simple, lograba entablar conversaciones con personas que creen que están hablando con otra persona. Este programa se considera el primer "*Chatbot*" de la historia y plantea el surgimiento del área del "*Procesamiento del Lenguaje Natural*".



Figura 40: John McCarthy



Figura 41: Frank Rosenblatt



Figura 42: Joseph Weizenbaum

De 1966 a 1972 fue desarrollado "Shakey" (figura 43), el cual se considera el primer robot de propósito general y que puede tomar sus propias decisiones de acuerdo a los datos que recopila de sus sensores. Los investigadores Charles Rosen, Nils Nilsson y Peter Hart lo construyeron en el "Stanford Research Institute". En 1979 unos estudiantes de la Universidad de Stanford crearon el "Stanford Cart" (figura 44), el cual era un vehículo autónomo que podía navegar y evadir obstáculos utilizando visión por computadora. Esa época fué de un gran auge inicial, donde el gobierno de Estados Unidos empezó a invertir en proyectos de IA (figura 45).

En 1972 James Lighthill, un matemático y profesor inglés, publicó el artículo "Artificial Intelligence: a paper symposium" (Lighthill, 1972), en el cual daba un panorama pesimista sobre el uso a futuro de la Inteligencia Artificial. Aunque fue publicado en 1973 su impacto fue muy fuerte durante toda la década de los 70's y parte de los 80's, iniciando con lo que se denominó el "Invierno de la Inteligencia Artificial". Muchos investigadores, incluyendo a algunos de los mismos fundadores del movimiento de Inteligencia Artificial, desestimaron su potencial, por lo cual se fueron cancelando muchos proyectos como por ejemplo los de procesamiento automático de lenguaje.



Figura 43: Robot Shakey



Figura 44: Stanford Cart



Figura 45: Gerald Ford supervisando



Material Adicional: Para consultar de manera mas detallada algunos eventos cronológicos sobre Machine Learning, se puede consultar una tabla muy completa de Wikipedia:

https://timelines.issarice.com/wiki/Timeline_of_machine_learning

Ocurrieron otros eventos históricos acerca del Machine Learning pero se tratarán en el siguiente capítulo por ser parte un poco más del área de Deep Learning.

6.2.2 Primeras Aplicaciones

Las técnicas y métodos que se utilizan en “*Machine Learning*” ya tiene algunos años, y algunas de las primeras aplicaciones que se empezaron a realizar fueron las de reconocimiento de caracteres OCR (“*Optical Character Recognition*” por sus siglas en inglés) y las de filtrado de correos indeseados (mas conocido por su término en inglés como “*Spam Filter*”) durante la década de 1990.

Inicialmente estas aplicaciones no dieron los mejores resultados e incluso se pensó que se tendrían que usar otras técnicas, pero con el incremento del uso de datos, los sistemas mejoraron y se empezaron aplicar a resolver problemas en otras áreas del conocimiento. Actualmente las técnicas de Machine Learning y especialmente las de Deep Learning (que se comentarán en el siguiente capítulo de este libro) son las más utilizadas y cada día tienen un mayor grado de eficiencia computacional y exactitud en sus resultados.

6.3 Clasificaciones de Algoritmos de Machine Learning

Dentro del área de Machine Learning, cada autor que ha aportado publicaciones e investigaciones proponen diferentes categorías o clasificaciones, pero las mas generales en que podemos clasificar al Machine Learning son las siguientes:

- Aprendizaje Supervisado.
- Aprendizaje No Supervisado.
- Aprendizaje Semi-Supervisado.
- Aprendizaje por Refuerzo.

Si hacemos una clasificación de acuerdo a los algoritmos empleados dentro del Machine Learning podemos clasificarlos de la siguiente manera:

- Regresión Lineal y Logística.
- Árboles de Decisión.
- Clasificadores Probabilísticos.
- K-Means.
- Support Vector Machines.
- KNN.
- Random Forest.
- Redes Neuronales.

No todos los autores están de acuerdo en estas clasificaciones, ya que algunos proponen otras categorías ó algunas de ellas son parte de alguna otra, pero éstas serán las que se manejarán en este libro por ser las más generales. Hay que decir que dentro de cada una de las divisiones anteriores existen subdivisiones y algunas podrían aparecer dentro de varias categorías a la vez, de acuerdo al objetivo que se decida para resolver un problema, mas adelante se comentará a detalle.

6.3.1 Aprendizaje Supervisado

Los sistemas de “*Aprendizaje Supervisado*” son aquellos donde los datos proporcionados viene debidamente etiquetados y clasificados, debido a esta característica se utiliza especialmente para los siguientes tipos de aplicaciones:

- Clasificación
- Regresión

Los métodos que utilizan “*Aprendizaje Supervisado*” se dividen en 2 partes, la primera consiste en “*entrenar*” el sistema, es decir, generar un conjunto de datos debidamente etiquetados y ordenados para que el sistema pueda “*aprender correctamente*”.

En el siguiente ejemplo (figura 46) recolectamos un conjunto de fotografías que han sido etiquetados con “*gato*” y con el “*nombre del gato*”. Con estos datos se crea un dataset, el cual será la entrada al algoritmo de entrenamiento para generar un modelo:

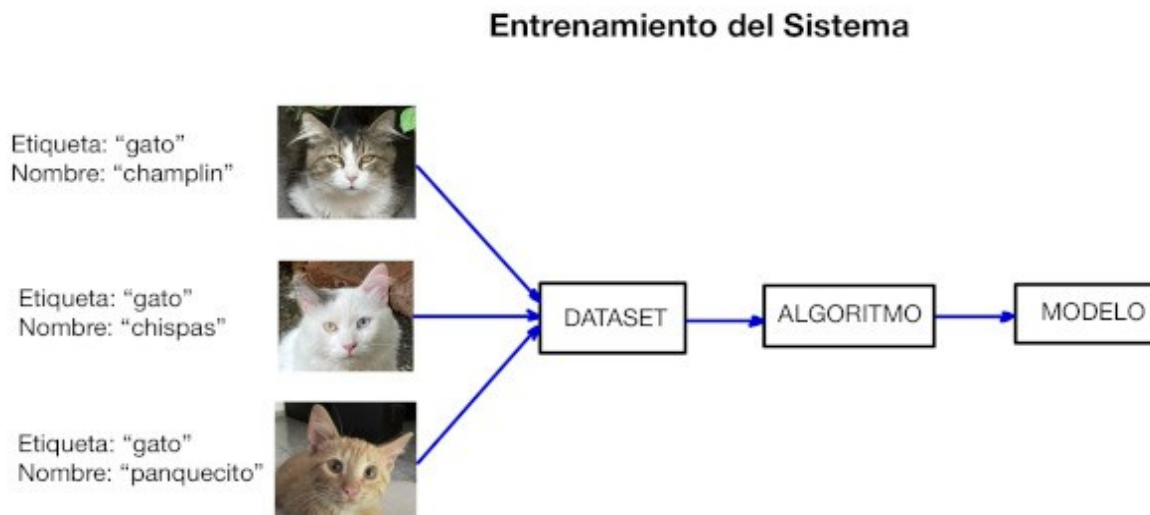


Figura 46: Entrenamiento del Sistema

Una vez que tenemos entrenado nuestro “*modelo*”, entonces ya podemos pasar a la parte de “*clasificación*”, donde debemos de contar con otro conjunto de pruebas, el cual debe de ser diferente del que se usó para el entrenamiento.

De este nuevo conjunto le damos 2 fotos nuevas para que el sistema nos diga si es ó no es un gato (recordar que el sistema sólo se entrenó para gatos), (figura 47).

En el primer intento nos indica que es “**ES un gato**”, y en el segundo que “**NO ES un gato**”, lo cual está correcto. Estos resultados dependen de haber realizado un buen entrenamiento con un conjunto bastante grande de imágenes.

Clasificación de Imágenes

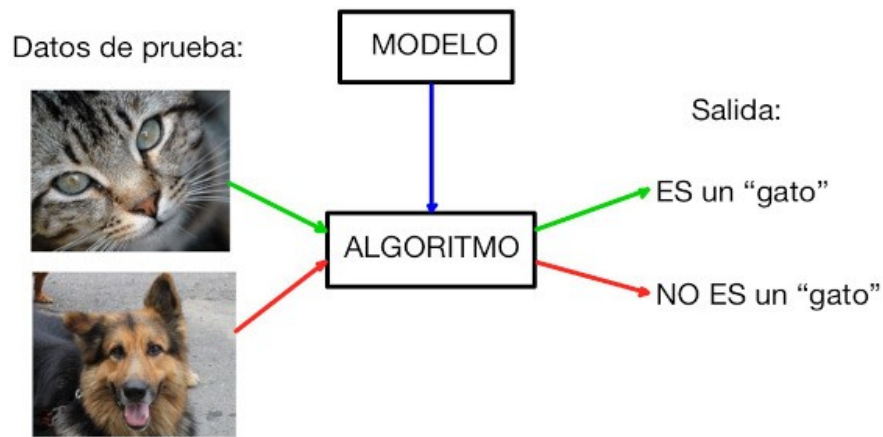


Figura 47: Clasificación de imágenes

El ejemplo anterior es usando imágenes, pero se pueden utilizar otros tipos de datos.

6.3.2 Aprendizaje No Supervisado

Los sistemas de “*Aprendizaje No Supervisado*” son aquellos donde los datos proporcionados no vienen etiquetados y el sistema trata de entender el conjunto de datos que se le dieron y busca ordenarlos o clasificarlos de acuerdo a cierta similitud entre ellos.

Son buenos para encontrar patrones que a veces no son fáciles de reconocer. También son buenos para detección de anomalías, es decir encontrar datos que se encuentran fuera de los patrones normales que siguen los demás datos.

También son buenos para encontrar y establecer reglas que suceden entre los datos, es decir, pueden detectar que siempre que sucede el evento "A" ocurre también "B", lo cual nos puede llevar a crear y generalizar reglas que podemos aplicar sobre los datos.

Un ejemplo general sería darle un conjunto de imágenes sin etiquetar a un algoritmo y que trate de juntarlos por similitud. En la figura 48 se observa que se introduce un conjunto de imágenes no etiquetadas a un algoritmo de clasificación el cual nos entrega a la salida la cantidad de Clases que encontró, donde una clase es un conjunto de datos que tienen cierta similitud entre ellos.

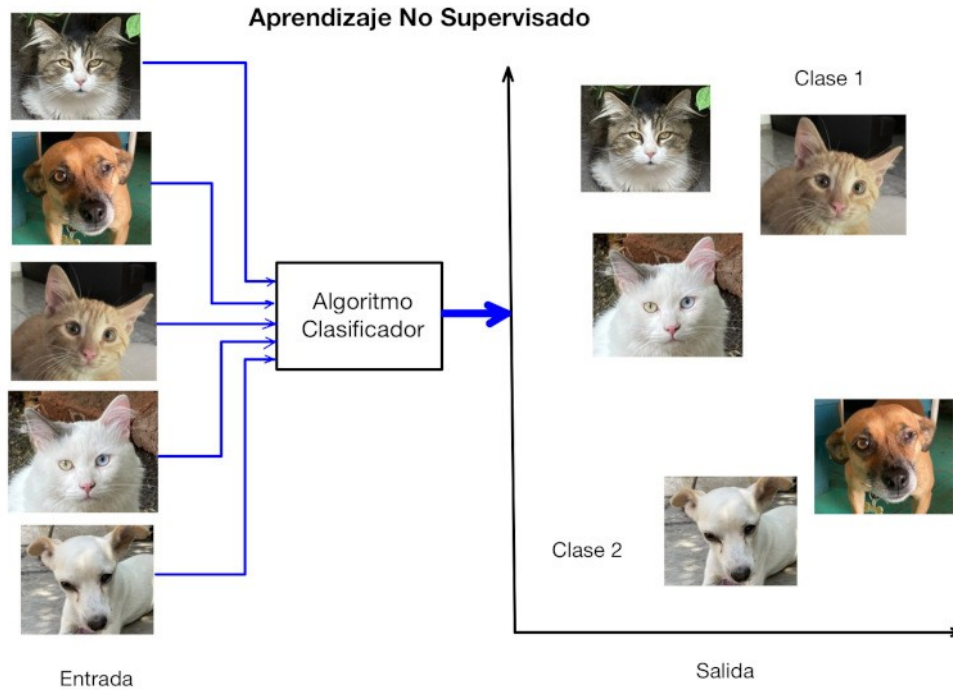


Figura 48: Aprendizaje No Supervisado

6.3.3 Aprendizaje Semi-Supervisado

Existe otro tipo de sistemas llamados “*Semi-Supervisados*”, donde se combinan ambas técnicas anteriores, ya que una parte de los datos están etiquetados y otra no, esto permite agrupar los datos con etiquetas y después identificar y agrupar a los otros datos.

6.3.4 Aprendizaje por Refuerzo

En el “*Aprendizaje por Refuerzo*”, a diferencia del “*Aprendizaje Supervisado y No Supervisado*”, no se tiene un conjunto de datos (dataset) a partir del cual partir. Ahora se utiliza una técnica en donde existe un agente encargado de explorar el ambiente y determinar las acciones que se tengan que hacer para lograr un objetivo por medio de prueba y error.

El sistema irá aprendiendo paulatinamente de acuerdo a las recompensas ó penalizaciones a partir de las acciones realizadas (Torres, 2021). Este tipo de aprendizaje se comentará más a profundidad en el siguiente capítulo.

6.4 Algoritmos de Machine Learning

6.4.1 Regresión Lineal

Es un algoritmo de “*Aprendizaje Supervisado*” que se utiliza principalmente para hacer predicciones. Emplea métodos estadísticos para estimar la relación entre una variable dependiente y una o más variables independientes.

La idea principal es dado un conjunto de datos, encontrar una aproximación lineal de su comportamiento y así poder estimar posibles valores nuevos. Ecuación utilizada:

$$y = mx + b \quad (38)$$

Donde:

- "y" es la variable dependiente.
- "m" es la pendiente de la recta.
- "x" es la variable independiente.
- "b" es un escalar.

La siguiente gráfica (figura 49) es un una regresión lineal que utiliza 50 puntos aleatorios con una Distribución Gaussiana alrededor de la línea:

$$y = 1.533858 x + 2.129333 \quad (39)$$

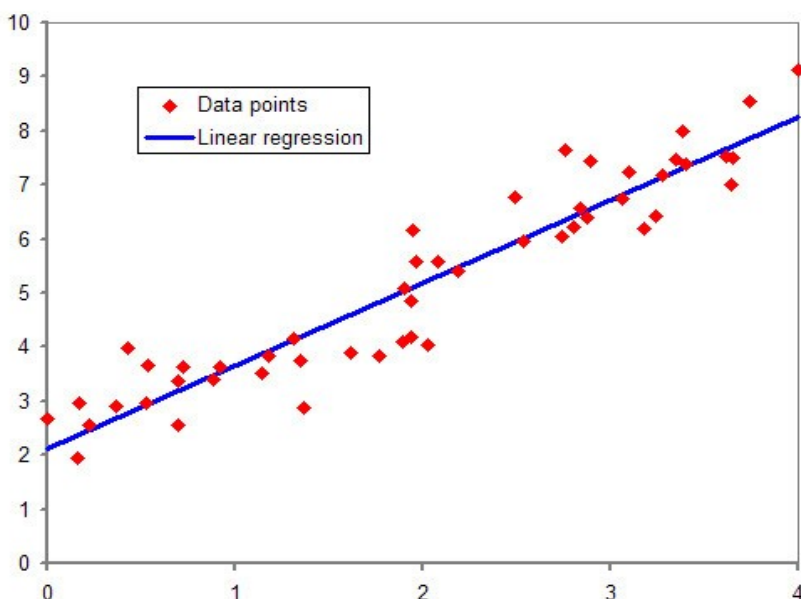


Figura 49: Regresión Lineal

Aplicaciones:

- Predicción del clima.
- Estimación de ventas.
- Optimización de precios de productos.

6.4.2 Regresión Logística

Es un algoritmo de "Aprendizaje Supervisado" que se utiliza principalmente para hacer clasificaciones binarias (figura 50). Es ampliamente utilizado cuando se tiene un problema, en donde dados ciertos datos de entrada, se tiene que clasificar como "verdadero" ó "falso". La siguiente gráfica es una función logística con:

$$B_0 + B_1 + e \quad (40)$$

en el eje horizontal y pi en el eje vertical:

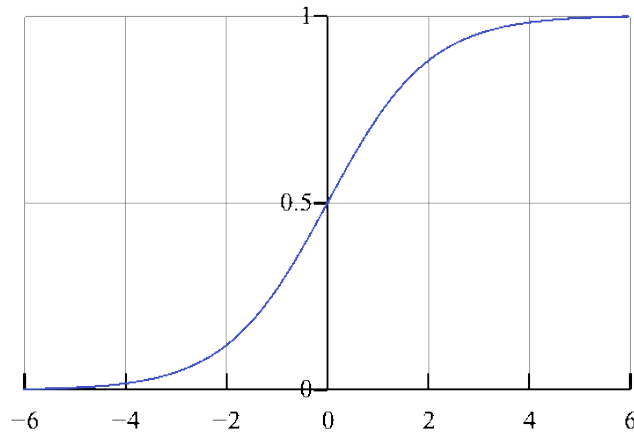


Figura 50: Regresión Logística

Aplicaciones:

- Reconocimiento de cáncer.
- Análisis crediticio.
- Predicción de temblores.

6.4.3 Árboles de Decisión

Es un algoritmo de "Aprendizaje Supervisado" que se utiliza principalmente para hacer predicciones de acuerdo a un conjunto dado de observaciones. Se utilizan "árboles de clasificación" en donde las hojas del árbol clases de etiquetas y las ramas representan conjunciones de características que llevan a clasificar a las clases de etiquetas. Se utilizan principalmente para la toma de decisiones ya que por su naturaleza gráfica resultan ser muy explícitos. El siguiente árbol (figura 51) muestra las características de los supervivientes al naufragio del "Titanic" (Wiki_0017, 2022), con los datos obtenidos de los pasajeros:

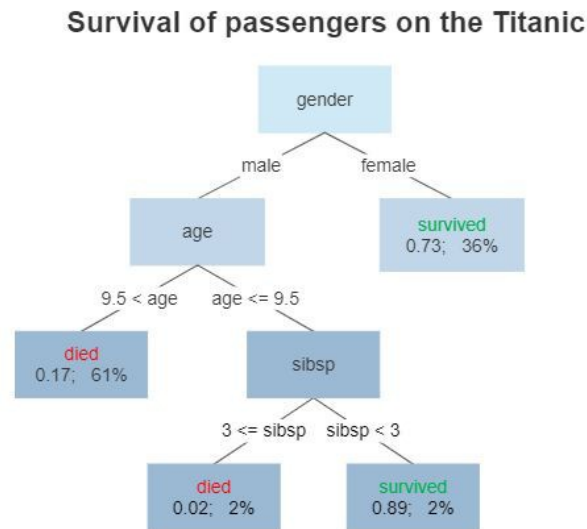


Figura 51: Árbol de Decisión

Aplicaciones:

- Elegir el vuelo adecuado para viajar.
- Sugerir el carro a comprar.

6.4.4 Clasificadores Probabilísticos

Es un conjunto de algoritmos que utilizan "*Aprendizaje Supervisado*" y que se utilizan principalmente para predicción. En este tipo de algoritmos, dado un conjunto de datos observados y de acuerdo a ciertas probabilidades, se puede predecir una posible salida.

Dentro de este conjunto de algoritmos, el más utilizado es aquellos que se basan en "*Redes Bayesianas*", que surgen a base del "*Teorema de Bayes*", publicado por el matemático inglés Thomas Bayes en 1763. La implementación más simple es el "*Naive Bayes*" (ó "*Clasificador Bayesiano Ingenuo*" en español).

Por ejemplo, se puede determinar si la hierba está mojada, de acuerdo a 2 posibles eventos, la lluvia y el agua del rociador. Anteriormente, en este mismo libro, en el Capítulo 5 (Probabilidad) se comentó acerca de las Redes Bayesianas (figura 32). Algunas de las aplicaciones mas comunes son:

- Reconocimiento de rostros.
- Detección de correo SPAM.
- Análisis de sentimientos.
- Sistemas de recomendación.
- Clasificación de artículos y textos.

6.4.5 Algoritmos de Agrupamiento: K-Means

Es un algoritmo de "*Aprendizaje No Supervisado*" que se utiliza principalmente para clasificación. De acuerdo a un conjunto dado de n datos, el algoritmo tratará de encontrar un conjunto de k agrupamientos (k -Clusters) que compartan ciertas características. Cuando se le asigne un caso de prueba, tratará de encontrar con cual de los k -Clusters tiene más afinidad, de acuerdo con la cercanía al promedio del conjunto de datos de cada uno de los k -Clusters.

En el siguiente ejemplo (figura 52) se observa un conjunto de datos que bajo el algoritmo "*K-Means*" (así se le denomina a este algoritmo) han sido encontrados 3 Clusters (agrupamientos):

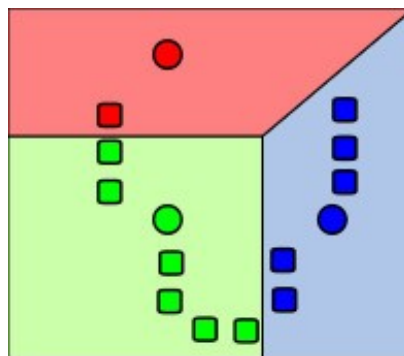


Figura 52: Clasificación con *K-Means*

Algunos ejemplos de aplicaciones de K-Means son::

- Elegir mejores rutas de transporte.
- Identificar noticias falsas.
- Clasificar libros de acuerdo a su género.
- Detección y filtrado de correo SPAM.

6.4.6 Support Vector Machines

Es un algoritmo de "*Aprendizaje Supervisado*" que se utiliza principalmente para clasificar y hacer análisis de regresión. Fue propuesto por Vladimir Vapnik de los Laboratorios Bell y que normalmente se abrevia como SVM (por sus siglas en inglés). Dado un conjunto de muestras para el entrenamiento, el SVM etiqueta las clases para que con una nueva muestra pueda predecir el tipo de clase a la que pertenece. A diferencia de otros algoritmos donde se trabaja sobre un mismo plano, el SVM genera hiperplanos con diferentes números de dimensiones, lo que permite clasificar de mejor manera algunos conjuntos de datos (figura 53).

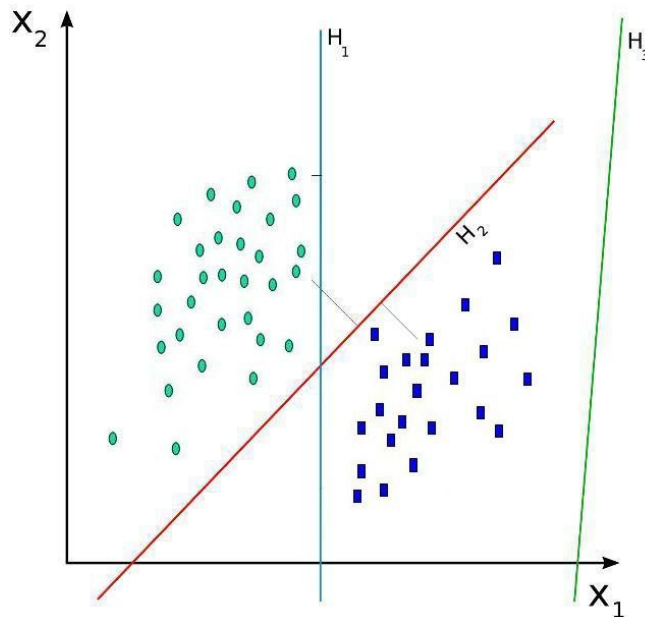


Figura 53: Hiperplanos con Support Vector Machine

Algunas de las aplicaciones de SVM son:

- Reconocimiento de escritura
- Clasificación de textos.
- Clasificación de imágenes.
- Clasificación de proteínas.

6.4.7 KNN

Es un algoritmo de "*Aprendizaje Supervisado*" que se utiliza principalmente para clasificación y regresión. El nombre completo del algoritmo es "*k vecinos más cercanos*", aunque es más conocido por su nombre en inglés "*k-nearest neighbors*" y para abreviar se usa más KNN (por sus siglas en

inglés). Este algoritmo fue desarrollado por Evelyn Fix y Joseph Hodges en 1951 y no debe de ser confundido con "K-Means", ya que son diferentes.

En este algoritmo primero se calcula la distancia de todos los datos en entrada por medio de una "función de distancia". Dado un dato de prueba y dependiendo del objetivo buscado (clasificación ó regresión) se puede hacer lo siguiente:

- Si es "Clasificación" se identifica el dato de entrada con la etiqueta mas frecuente.
- Si es "Regresión" se calcula la salida de acuerdo al valor promedio de los vecinos cercanos.

En la figura 54 se observa que el algoritmo KNN tratará de identificar si la figura verde (círculo) pertenece a las figuras rojas (triángulos) ó a las figura azules (cuadros).

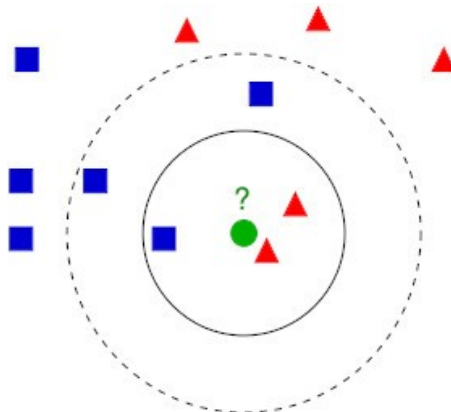


Figura 54: Algoritmo KNN

Aplicaciones:

- Delitos bancarios.
- Análisis en el mercado de valores.
- Detección de huellas digitales.

6.4.8 Random Forest

Es un algoritmo de "Aprendizaje Supervisado" que se utiliza principalmente para predecir o escoger mejores opciones. Al igual que el método anterior utiliza árboles de decisión, aunque ahora genera varios árboles con características diferentes para encontrar la mejor opción (figura 55).

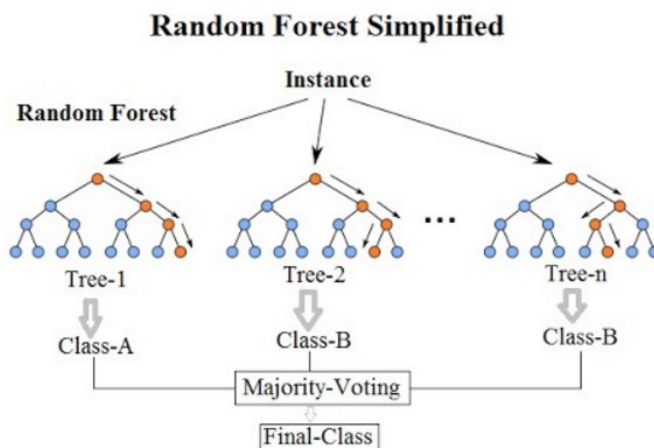


Figura 55: Random Forest

En la figura anterior se observan 3 árboles de decisión que se generan y de los cuales se seleccionará el que mejores resultados aporte. Algunas de aplicaciones mas comunes son:

- Identificar enfermedades analizando historiales médicos.
- Detección de fraudes en bancos.
- Estimar pérdidas y ganancias en mercados de valores.

6.4.9 Redes Neuronales

Las “Redes Neuronales” se analizarán a mas detalle en el siguiente capítulo y es la técnica que evolucionó a lo que se denominó “Deep Learning”, la cual es el área de mayor crecimiento y aplicación en la actualidad.

6.5 Empezando con Machine Learning

Dentro del área de Machine Learning no existe un solo algoritmo que se pueda aplicar en todos los casos. Actualmente, las Redes Neuronales han cobrado mucho auge, pero eso no significa que por medio de ellas se pueda solucionar cualquier cosas. El investigador inglés Geoffrey Hinton (impulsor y creador del Deep Learning) ha afirmado que “*el Deep Learning será capaz de hacer cualquier cosa*” (Hao, 2020) .

Por lo tanto, para obtener los mejores resultados, es preferible hace un banco de pruebas con varios algoritmos y escoger el que mejor resultados consiga. Dependiendo del tipo de problema es posible un algoritmo que se adecúe más que otros, y esto solo lo proporciona el conocimiento que se tenga de los algoritmos y la experiencia al resolver muchos casos.

6.5.1 Datasets

En el capítulo 2 se comentó acerca de los “*datasets*”, sus características y algunas recomendaciones de cómo deberían estar estructurados. Para iniciar con los procesos de Machine Learning se requiere de “*datos*”, entre más, mejor, pero sin perder la calidad. En el Anexo 10 se encuentra una lista de sitios en Internet de donde se pueden descargar de manera abierta y gratuita datasets.



Advertencia: Se recomienda descargar los datasets de los sitios oficiales, ya que existen muchos que se pueden descargar de otros lugares pero pueden tener información alterada.

Algunos de los ejemplos propuestos que vienen en este libro utilizan “*Scikit-Learn*”, que es una biblioteca gratuita para Python. Contiene funciones para realizar clasificación, regresión, agrupamiento (clustering), y que incluye algoritmos para trabajar con Support-Vector Machine, Random Forest, Potenciación del Gradiente, K-Means y DBSCAN. Lo anterior hace que Scikit-Learn sea una de las bibliotecas más usadas en Machine Learning.



Material Adicional: Para mayor información se puede consultar la página oficial de Scikit-Learn: <https://scikit-learn.org/stable/>

Dentro de “*Scikit-Learn*” se encuentra un conjunto de datasets con los cuales se puede iniciar a programar, los cuales fueron clasificados de 2 tipos (de acuerdo a su complejidad):

- Toys (sencillos):

- Boston house prices.
- Iris plants.
- Diabetes.
- Optical recognition of handwritten digits.
- Linnerrud.
- Wine recognition.
- Breast cancer wisconsin (diagnostic).
- Real World (complejos):
 - The Olivetti faces.
 - The 20 newsgroups text.
 - The Labeled Faces in the Wild face recognition.
 - Forest covertypes.
 - RCV1.
 - Kddcup 99.
 - California Housing.



Material Adicional: Se puede consultar las características específicas de cada uno de los datasets anteriores en: <https://scikit-learn.org/stable/datasets.html>

6.5.2 Entrenamiento

Muchos sistemas han adoptado que todos los datos que se tienen, el 80% se asigna al entrenamiento del modelo y el otro 20% se usa para realizar las pruebas, aunque si el conjunto de datos es muy grande, se puede asignar un porcentaje menor para hacer las pruebas (se llega a veces hasta el 1%).

En 1996 David Wolpert publica el artículo "*The Lack of A Priori Distinctions Between Learning Algorithms*", donde propone que si no se tiene información específica sobre los datos, no se debe tener preferencia por uno u otro modelo.

Muchos sistemas han adoptado que todos los datos que se tienen, el 80% se asigna al entrenamiento del modelo.

6.5.1 Sobreajuste y Subajuste

El "*Sobreajuste*" ó "*Overfitting*" (por su término en inglés) ocurre cuando a un modelo se le entrena con suficientes datos pero aún así no logra generalizar correctamente. Posibles formas de solucionar este problema:

- Simplificar el modelo seleccionando menos parámetros.
- Reducir el ruido en el entrenamiento.

El "*Subajuste*" ó "*Underfitting*" (por su término en inglés es el contrario de "*Overfitting*" y es cuando el modelo es muy simple para poder aprender la estructura de los datos. Para solucionar esto:

- Cambiar de modelo con mas parámetros.

6.6 Aplicaciones con Machine Learning

A continuación viene algunos ejemplos que utilizan diferentes algoritmos de Machine Learning (de los mas sencillos a los mas complejos). Es importante hacer notar que de acuerdo al objetivo que se busque, se debe proponer un algoritmo a utilizar y no al revés.

Algunos ejemplos podrían resolverse de diferentes maneras, ya sea ajustando un mismo algoritmo ó utilizando varios algoritmos, por lo que se tiene que valorar el que mejor resultado nos arroje. El seleccionar un “*buen algoritmo*” para resolver un problema es algo que lleva tiempo, experiencia y conocimiento del área de Ciencia de Datos.

6.6.1 Regresión lineal



Actividad: Predecir la temperatura promedio para julio del año 2022 en Nueva York.

Para este ejemplo utilizaremos “*Aprendizaje Supervisado*” “con “*Regresión Lineal*” por su carácter predictivo. Primero necesitamos el dataset adecuado, que contenga la información climática promedio de la ciudad de Nueva York durante los últimos años (se propone de 1985 hasta el año 2021). Para descargar esta información entramos al sitio: “*Climate at a Glance*”:

- <https://www.ncei.noaa.gov/access/monitoring/climate-at-a-glance/>

Y seleccionar:

- City

Ahora seleccionamos los siguientes datos:

- Parameter: Average Temperature
- Time Scale: 1-Month
- Month: july
- Start Year: 1985
- End Year: 2022
- State: New York
- City: New York

Hacemos click en:

- Plot

Y aparece en la parte inferior una gráfica de temperaturas promedio con su respectiva tabla de datos. Para descargar los datos hacemos click en el icono de Excel (para formato CSV) que se encuentra a la derecha de la palabra:

- Download

Y se descarga el archivo:

- *USH00305801-tavg-1-7-1985-2022.csv*

Abrimos el archivo y se observan los primeros 4 renglones del dataset que son los siguientes:

- New York, New York, Average Temperature, July
- Units: Degrees Fahrenheit
- Base Period: 1901-2000
- Missing: -99

Estos renglones los borramos manualmente para poder procesar correctamente el archivo CSV y ahora los primeros 4 renglones nos quedaran como sigue:

- Date, Value, Anomaly
- 198507, 75.3, -0.1

- 198607,75.2,-0.2
- 198707,77.1,1.7
-

De los cuales se observa que el primero es la cabecera de nuestros datos CSV, y los 3 siguientes renglones son las primeras 3 observaciones, las cuales empiezan por el año "1985" seguida de un "07" que equivale al mes 7 (julio), luego viene la primer temperatura "75.3" (en grados Fahrenheit) y por último la anomalía "-0.1", que nos indica si ese mes estuvo por debajo (signo negativo) o por encima del promedio calculado para toda la serie que va desde 1901 hasta el año 2000.

Para no modificar el archivo original, este nuevo archivo lo grabamos como:

- *nueva_york_julio_1985-2022.csv*

A continuación se observan los primeros 10 renglones de nuestro dataset (el archivo completo contiene 38 renglones incluyendo la cabecera):

```
nueva_york_julio_1985-2022.csv X
1 Date,Value,Anomaly
2 198507,75.3,-0.1
3 198607,75.2,-0.2
4 198707,77.1,1.7
5 198807,78.5,3.1
6 198907,74.1,-1.3
7 199007,75.9,0.5
8 199107,76.9,1.5
9 199207,73.3,-2.1
10 199307,79.3,3.9
```

Este será el dataset con el que vamos a trabajar. A continuación pasaremos con el código en Python empezando por cargar las bibliotecas necesarias. Después cargamos también el dataset (que descargamos y procesamos anteriormente) y lo convertimos a un dataframe usando Pandas:

```
1 # Cargar bibliotecas necesarias
2 import pandas as pd
3 from scipy import stats
4 # Cargar datos
5 print("\nCargando datos")
6 df_nueva_york = pd.read_csv('nueva_york_julio_1985-2022.csv')
```

Ahora cambiamos el nombre de las columnas de inglés a español para hacerlo mas legible y eliminamos el "07" (que indica el mes 7, o sea julio) que aparece en la primer columna de cada registro, dividiendo cada elemento entre 100:

```
7 # Cambiando el nombre de las columnas
8 df_nueva_york.columns = ['Fecha', 'Temperatura', 'Anomalía']
9 # Eliminamos el mes (07) dividiendo cada fecha entre 100
10 df_nueva_york.Fecha = df_nueva_york.Fecha.floordiv(100)
```

Ahora convertimos cada uno de los elementos de la segundo columna, de Fahrenheit a grados centígrados, para ello recorreremos todo el dataset y modificamos los datos aplicando la fórmula de

conversión adecuada. Después imprimimos el dataset completo ya completo con los cambios que hemos aplicado (solo para corroborar) e imprimimos su valores estadísticos de todos los datos (promedio, desviación estándar, mínimo, máximo) por medio de la función “describe”:

```
11 # Convertir Farenheit a Grados Centígrados
12 for i in df_nueva_york.index:
13     df_nueva_york.Temperatura[i] = 5 / 9 * (df_nueva_york.Temperatura[i]-32)
14 # Imprimir dataset completo con los cambios
15 print("\nDataset completo ya con cambios:\n",df_nueva_york)
16 # Imprimir valores estadísticos
17 print("\nValores Estadísticos del Dataset:\n",df_nueva_york.Temperatura.describe())
```

Ahora viene el cálculo de la predicción de la temperatura, para lo cual utilizamos regresión lineal, calculando la pendiente *m* (con la función “*slope*”), a “*x*” le asignamos 2022 (que es el año para el que queremos calcular) y calculamos la constante “*b*” (con la función “*intercept*”). Ya que tenemos todo, calculamos el punto que queremos usando la función $y=mx+b$:

```
18 # Predecir temperatura de Nueva York en julio del 2022
19 # Usando:  $y = mx + b$ 
20 regresion_lineal = stats.linregress(x=df_nueva_york.Fecha, y=df_nueva_york.Temperatura)
21 m = regresion_lineal.slope
22 x = 2022
23 b = regresion_lineal.intercept
24 estimacion = m * x + b
25 print("\nLa temperatura media estimada para Nueva York en Julio del 2022 es: ", estimacion, "\n")
```

Por último observamos impreso en pantalla nuestra predicción de temperatura promedio para el mes de julio del año 2022 en la ciudad de Nueva York:

```
La temperatura media estimada para Nueva York en Julio del 2022 es: 25.796296296296305
```


Capítulo

Deep Learning

7

Objetivo: Conocer la evolución y aplicación de los diferentes tipos actuales de Deep Learning

“Creo que el Deep Learning lo podrá hacer todo!”

Geoffrey Hinton

El Inglés Geoffrey Hinton es actualmente Profesor Investigador de la Universidad de Toronto y propuso el uso del algoritmo "Back Propagation" en redes neuronales multicapa. En el 2006 acuña el término "Deep Learning" (Aprendizaje Profundo) a esta nueva arquitectura y en 2012 lo aplicó para reconocimiento de imágenes logrando resultados sobresalientes, impulsando fuertemente la investigación en este campo, por lo cual se le reconoce como el "Padre del Deep Learning".



Continuando con el área de Machine Learning, este capítulo se adentrará primero en la parte de las Redes Neuronales, para después pasar a las *“Redes Neuronales Profundas”*, o de varias capas, mejor conocidas como *“Deep Learning”*, para después comentar los últimos avances en esta área.

7.1 Conceptos

7.1.1 Definición

No existe como tal una definición específica para el *“Deep Learning”*, ya que esta área empezó tomando algunos algoritmos de Machine Learning (específicamente las redes neuronales), a las cuales se les agregó varias capas intermedias en cascada, de ahí su nombre de *“Deep”* (profundidad en español). El científico e investigador inglés Geoffrey Hinton fue el primero de asignarles el nombre de *“Deep”*, de ahí que muchos autores lo consideran como *“Padre del Deep Learning”*.

Al Deep Learning en idioma español se le conoce como *“Aprendizaje Profundo”*, pero es mas conocido en todo el mundo como *“Deep Learning”*, por lo que será el nombre que se utilizará durante todo este libro.

7.2 Historia

7.2.1 Primeros desarrollos

Como se habló en el capítulo anterior, el "*Invierno de la IA*" hizo que bajara el interés y el desarrollo de esta área, por lo cual durante varios años se prefirió investigar en otros campos de la ciencia. Algunos autores sitúan un solo "*invierno*", mientras que otros comentan acerca de 2 "*inviernos*" (1974 a 1980 y de 1987 a 1993), lo cierto es fue una época que afectó el avance de la IA durante los años 70's y 80's. Fue hasta los 90's cuando nuevos avances e investigaciones hicieron que se retomara nuevamente esta área.

Mientras la Inteligencia Artificial se encontraba en un "*Invierno*" durante la década de 1970 y parte de 1980, se inició un cambio importante en tecnología con la aparición el 1 de junio de 1979 del microprocesador Intel 8088 (figura 56), el cual marcaría el inicio de la computación personal (figura 57) y el crecimiento del hardware para procesamiento. Este chip costaba \$124.80 dólares en Estados Unidos, con una frecuencia de reloj de 5MHz y un bus de datos de 8 bits.

En 1975 Cray Research lanza al mercado la computadora CRAY-1 (figura 58), la cual se considera la primer supercomputadora, iniciando el camino a la creación de los grandes centros de cálculo y datos que hoy conocemos. Esta computadora tenía un costo de \$7.9 millones de dólares en Estados Unidos, con un procesador de 64 bits y 80 MHz, una memoria de 8.39 Megabytes, un almacenamiento de 303 Megabytes, un peso de 5.5 toneladas, un consumo de energía de 115 KW y un desempeño de hasta 160 MegaFlops.



Figura 56: Intel 8088



Figura 57: IBM Personal Computer



Figura 58: Cray 1

En 1995 IBM construye una supercomputadora únicamente con el fin de convertirse en experta en jugar ajedrez a la que denominó "*Deep Blue*" (figura 58). Utilizaba la arquitectura RS/600 de IBM y en su versión final tenía 30 procesadores PowerPC de 200 MHz y otros 480 procesadores VLSI especializados en ajedrez, logrando 11.38 GigaFlops de desempeño.

En 1996 "*Deep Blue*" se enfrentó a Garry Kasparov (figura 60), que en ese momento era campeón del mundo en ajedrez, en un match de ajedrez en la ciudad de Filadelfia en Pensilvania y fue vencida la máquina. Al año siguiente, del 3 al 11 de mayo de 1997 se llevó la revancha en la Ciudad de Nueva York, pero esta vez la "*Deep Blue*" ganó el match. Esto marca un hito importante en la historia, ya que era la primera vez que una máquina le ganaba a un hombre en ajedrez. Muchos investigadores e

intelectuales pensaban que nunca se le podría ganar en ajedrez a un hombre lo que ocasionó ver a las máquinas con la capacidad de lograr capacidades hasta ese momento desconocidas y que marca nuevamente el repunte del área de Inteligencia Artificial.

En 2006 Geoffrey Hinton (figura 61) publica el artículo "*A Fast Learning Algorithm for Deep Belief Nets*", el cual logra reconocer dígitos escritos a mano con una precisión de mayor a 98%. Para ello utilizó una red neuronal de varias capas, y a esta técnica le denominó "*Deep Learning*" (Hinton et al, 2006). El desempeño del algoritmo fue tal que impulsó del desarrollo del Deep Learning el cual continúa hasta nuestros días.



Figura 59: IBM Deep Blue



Figura 60: Gary Kasparov



Figura 61: Geoffrey Hinton

Después de "*Deep Blue*", IBM construye otra computadora para competir en el juego de "*Jeopardy*" contra humanos (figura 62). En febrero del 2011, durante 3 días, compitió contra Brad Cutter (mayor ganador de dinero hasta ese momento) y contra Ken Jennings (récord por la racha mas larga de juegos ganados) derrotándolos a ambos en un juego que también se había pensado no podría ganar una computadora.

"*Google Deepmind*" (que es parte de la compañía Google) también desarrolló una computadora especial para competir contra humanos en el juego de mesa "*Go*", a la cual denominó "*AlphaGo*". En marzo del 2016, se enfrentó al entonces campeón mundial Lee Sedol (figura 63), al cual derrotó por 4-1.

Google siguió desarrollando sus algoritmos e infraestructura (figura 64) y el 11 de marzo del 2022 sale publicado en el Washington Post un artículo donde Blake Lemoine (ingeniero de Google) afirma le hizo varias pruebas (a través de conversaciones) al chatbot inteligente "*LaMDA*" (desarrollado por Google). Después de realizadas las pruebas (junto con un colega suyo que también trabaja en Google, pero se mantuvo en el anonimato) afirma que *LaMDA* tiene "*vida y sentimientos propios*". Lemoine afirma que el programa se comporta como un niño de 7 ú 8 años que sabe de física y muchas cosas más (Tiku, 2022).

Este último evento marca nuevamente el debate sobre los límites y alcances de la Inteligencia Artificial y el gran avance que se ha logrado en este campo.



Material Adicional: La entrevista completa entre Blake Lemoine con *LaMDA* se puede consultar en Internet en:

<https://cajundiscordian.medium.com/is-lamda-sentient-an-interview-ea64d916d917>



Figura 62: IBM Watson



Figura 63: Lee Sedol



Figura 64: Datacenter de Google



Material Adicional: Para ver de manera gráfica algunos eventos cronológicos sobre Machine Learning, se puede consultar también este sitio:

<https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History>

7.2.2 Aplicaciones e Impacto Actual

Se puede decir que a partir de que en el año 2006 Geoffrey Hinton demostró que las redes neuronales de varias capas (que denominó Deep Learning y de ahí su nombre) eran capaces de lograr mejores resultados para el reconocimiento de números (con la ayuda de gran cantidad de datos) surge la “Era del Deep Learning”. Con este enfoque se empezaron a recolectar gran cantidad de datos que se fueron integrando junto con los algoritmos para poder lograr resultados que años antes parecían impensables de lograr. Día a día crecen las aplicaciones que utilizan Deep Learning, algunas de ellas son las siguientes:

- Reconocimiento y monitoreo de objetos.
- Sistemas de recomendaciones (películas, libros, música, etc.).
- Detección de patrones de consumo.
- Predicción de eventos naturales.
- Detección y prevención de enfermedades.
- Descubrimiento de nuevos componentes biológicos y materiales.

El auge de del Deep learning se dió por los siguientes avances:

- Mejores algoritmos de Machine Learning y Deep Learning
- Almacenamiento de gran cantidad de datos.
- Incremento en el poder de cómputo.

El impacto mundial del Deep Learning ha sido tan grande que ha puesto nuevamente a la Inteligencia Artificial como el punto de partida para el desarrollo de nuevas aplicaciones que se vayan a desarrollar en cualquier área del conocimiento humano.

7.3 Clases de Deep Learning

Aunque no existe una clasificación como tal dentro del Deep Learning, se hizo una lista con algunas de las técnicas de Deep Learning, desde la mas simple que es la “Red Neurona” (y la cual es el

punto de partida) hasta los enfoque mas nuevos que siguen surgiendo como son los “Transformers” y los nuevos enfoques y paradigmas donde se aplican varias técnicas como lo es la “Neuroevolución”. Por lo anterior, la lista nos queda de la siguiente manera:

- Redes Neuronales.
- Redes Neuronales Profundas (Deep Learning).
- Redes Neuronales Convolucionales.
- Redes Adversarias Generativas (GAN's).
- Aprendizaje por Refuerzo Profundo.
- Transformers.
- Neuroevolución.

7.4 Redes Neuronales

En el proceso para entender el funcionamiento del cerebro humano, se fueron desarrollando investigaciones y teorías acerca de su funcionamiento, dando un primer paso con el desarrollo de un modelo de una neurona artificial. Al conectar varias entre ellas se convirtió en lo que conocemos como “Red Neuronal Artificial” ó “Artificial Neural Network” (“ANN” por sus siglas en inglés). Una Red Neuronal Artificial (figura 65) trata de simular los procesos de sinapsis que ocurren las neuronas del cerebro creando un modelo matemático con entradas conectadas a una primer capa de neuronas, después viene una o varias capas intermedias y finalmente una capa de salida.

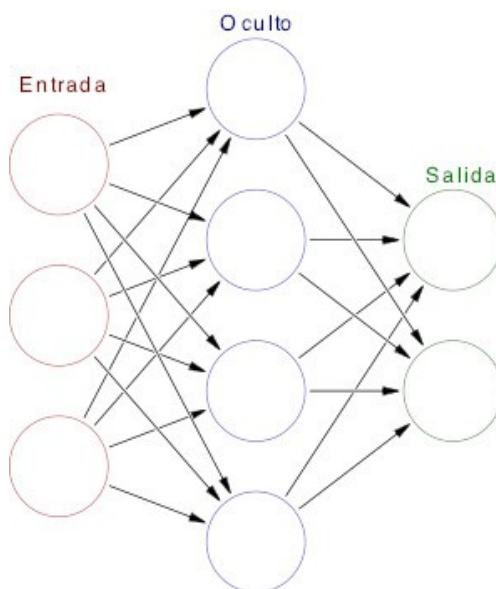


Figura 65: Red Neuronal Artificial

La señal de entrada se conecta a la red neuronal y entre cada conexión se va modificando su valor de acuerdo a los pesos que se les otorgue a cada una de las conexiones ó capas involucradas en el proceso. Al final se tiene una capa con valores diferentes en cual se traduce en la respuesta del sistema, en donde se puede colocar por ejemplo una imagen de entrada (por ejemplo de números escritos a mano) y a la salida nos puede indicar (por medio de valores de salida en la capa final) el número que se ha introducido en el sistema. Para lograr lo anterior y que la respuesta sea adecuada, se debe “entrenar” el sistema con una gran cantidad de datos, es decir, se debe colocar (para el ejemplo anterior de números) una por una las imágenes de muchos números ya etiquetados con su respectiva salida para que el sistema pueda ir calculado el valor de los pesos que se tienen que

asignar en cada una de las capas y de las conexiones de las neuronas. Este proceso se repite con cientos o miles de imágenes y por medio de una función de activación calcular su valores, y en caso de existir un error, tratar de minimizarlo al emplear funciones de retro-propagación ("*Back Propagation*"), hasta que el error desaparezca o al menos disminuya.

7.5 Redes Neuronales Profundas (Deep Learning)

Como se mencionó al principio de este capítulo, no existe como tal una definición para "*Deep Learning*", lo que se hizo fue ir agregando mas capas intermedias a un modelo básico de una red neuronal y a este tipo de arquitectura se le denominó "*Deep Learning*".

7.6 Redes Neuronales Convolucionales

Es un tipo de Red Neuronal Artificial que se modela muy parecido a un "*Perceptrón Multicapa*", en donde se su funcionamiento es muy parecido a como se desarrolla de manera biológica en la corteza visual del cerebro para detectar formas y figuras. Por lo anterior este tipo de redes neuronales artificiales son muy utilizadas para aplicaciones de visión artificial, como podría ser la detección, segmentación y clasificación de objetos dentro de una imagen.

7.7 Redes Adversarias Generativas (GAN's)

Las "*Redes Adversarias Generativas ó Antagónicas*", mas conocidas como GAN's ("*Generative Adversarial Network*" por sus siglas en inglés) es un sistema de Machine Learning compuesto por 2 redes neuronales, que trabajan (ó compiten entre ellas) en un juego denominado de "*suma cero*". Fue desarrollada por el científico Estadounidense Ian Goodfellow (figura 66) en 2014. En este tipo de arquitectura, una red neuronal se encarga de generar datos mientras la otra los evalúa o discrimina. Este tipo de sistemas han dado buenos resultados cuando se utilizan grandes cantidades de datos ya que el uso de la red discriminatoria permite obtener una mejor salida de datos. Actualmente son muy usadas para crear imágenes nuevas a partir de un conjunto de datos, por ejemplo para crear paisajes que no existen (dado miles de fotos de paisajes), como es el caso del cuadro "*A Recent Entrance to Paradise*", creado por el software "*Creativity Machine*" (figura 67) del Dr. Thaler. También pueden generar rostros ó caras de personas que no existen (a partir de miles de fotos de personas reales), como es el caso del software que genera rostros de mujer, desarrollado por bod Inga klang (figura 68).



Figura 66: Ian Goodfellow



Figura 67: Cuadro de Creativity Machine



Figura 68: Mujer generada con GAN



Material Adicional: Cada vez que se entra a este sitio, un algoritmo GAN (desarrollado por Tero Karras, un investigador de NVIDIA Research) genera un nuevo rostro:
<https://thispersondoesnotexist.com/>



Advertencia: Con GAN's se pueden crear situaciones en video que nunca existieron (llamados deepfake), los cuales pueden ser usados de manera positiva ó negativa. En este video revivieron a Cantinflas: <https://www.youtube.com/watch?v=dVycTJwEKa4>

7.8 Aprendizaje por Refuerzo Profundo

7.8.1 Conceptos sobre Aprendizaje por Refuerzo Profundo

El “*Aprendizaje por Refuerzo*” puede utilizar cualquier técnica empleada en Machine Learning, pero últimamente se ha desarrollado más el área de “*Aprendizaje por Refuerzo Profundo*” (derivado del Deep Learning, de ahí su nombre de profundo). Esta técnica está basada en el uso de redes neuronales con muchas capas de profundidad y algunas de las aplicaciones más avanzadas como AlphaGo (campeón de Go), AlphaStar (campeón de StarCraft II) ó Dota-2 (campeón de juegos electrónicos) utilizan esta técnica.

7.8.2 Componentes del Aprendizaje por Refuerzo Profundo

Algunos de los elementos de los que se compone el Aprendizaje por Refuerzo Profundo son (entre paréntesis su nombre mas común usado en inglés, ya que la mayoría de las publicaciones utilizan esos nombres):

- Tarea (Task).
- Agente (Agent).
- Entorno (Environment).
- Estado (State).
- Acción (Action).
- Función de Transición (Transition Function).
- Recompensa (Reward).
- Función de Recompensa (Reward Function).
- Iteración (Time Step).
- Episodio (Episode).

Tarea es el objetivo o meta a alcanzar.

Agente es el programa encargado de tomar decisiones con el objetivo de resolver un problema.

Entorno es todo aquello que nos representa el problema y que puede ser cambiante. Cuando se tiene una representación completa del entorno se puede encontrar la solución óptima. Sí la representación es incompleta el agente tendrá que tomar decisiones para poder encontrar alguna solución.

Estado es el conjunto de variables que representan al entorno y que nos indican la situación actual del entorno.

Acción es la operación a realizar por el Agente de acuerdo a la situación actual del Entorno y su Estado, a su vez, cada acción nos lleva a cambio de estado en el sistema.

Función de Transición es la función que se encarga de mapear el cambio entre estados.

Recompensa es lo que se obtiene cuando una buena Acción ejecutada por el Agente no acerca al objetivo propuesto

Función de Recompensa es la que se encarga de evaluar las acciones realizadas por el Agente para determinar las Recompensas que se asignen.

Iteración es cada uno de los ciclos que ocurren entre el Agente y el Entorno, donde el objetivo es acercarnos a la meta en cada iteración

Episodio es el conjunto de todas las iteraciones hasta llegar a la meta final el cual se denomina "*Tareas Episódicas*", pero si no existe un final como tal se denominan "*Tareas Continuas*".

7.9 Transformers

7.9.1 Conceptos

Un "*Transformer*" es un modelo de Deep Learning basada en "*mecanismos de atención*", prescindiendo por completo de la recurrencia y de los algoritmos de Convolución. Fue propuesta por el grupo Google Brain (que es parte de Google) cuando se publicó el artículo "*Attention Is All You Need*" en 2017 (Vaswani et al., 2017).

En esta técnica, dado un conjunto de entrenamiento el sistema puede generar nuevos datos manteniendo la misma estadística, es decir, si al sistema se le proporciona un conjunto de fotos por ejemplo de paisajes, puede, a partir de esos datos, crear un nuevo paisaje que no existe previamente. Actualmente los "*Transformers*" es la técnica mas utilizada para trabajar con "*Procesamiento del Lenguaje Natural*" y "*Procesamiento de Imágenes*", debido a sus resultados sobresalientes en estas áreas.

7.10 Tendencias

La complejidad y tamaño de las Redes Neuronal Artificiales sigue en aumento, por lo tanto, se han empezado a experimentar con algunas técnicas nuevas para mejorar su desempeño y mas que todo para lograr nuevos objetivos mas complejos. A continuación se comentarán algunas de las últimas tendencias que se están desarrollando.

7.10.1 Neuroevolución

Una Red Neuronal Artificial tiene la gran ventaja de que puede "*aprender*" (ó "*memorizar*" cómo lo dicen algunos investigadores) un cierto patrón sobre el cual fue entrenada, pero si algunas de las condiciones del sistema cambian, entonces es necesario volver a cambiar la estructura de la red (cantidad de capas intermedias y sus respectivos valores), así como posiblemente sus entradas y salidas (y las funciones de cálculo de todos sus pesos así como su función de activación).

Esto nos da por consiguiente que no sean muy flexibles, por lo que se ha buscado la forma de poderse adaptar de manera dinámica a los cambios y de ahí surge la idea de la "*Neuroevolución*". En esta técnica se aplica la "*Computación Evolutiva*" (que utiliza "*Algoritmos Genéticos*") para poder adaptarse a los cambios que se produzcan en el sistema y que ayude a la construcción de la red neuronal de manera dinámica.

De esta manera, los "*Algoritmos Genéticos*" se dedican a construir la estructura de la Red Neuronal, la cual posteriormente será entrenada y puesta en función. Todos estos procesos se hacen de manera dinámica, es decir, si existen cambios en cuanto al objetivo a conseguir por el sistema, nuevamente se tienen que calcular la red neuronal y volverse a entrenar, por lo que se consigue un

sistema mas dinámico y adaptable, utilizando los paradigmas de la “Computación Evolutiva” y la “Computación basada en Redes Neuronales” al mismo tiempo.

7.11 Aplicaciones con Deep Learning

A continuación se explicarán varios ejemplos resueltos y con código que utilizan técnicas de Deep Learning, empezando con los sencillos hasta algunos más complejos.

7.11.1 Reconocimiento de Números

Un primer ejemplo a trabajar será el siguiente:



Actividad: Reconocer números escritos a mano.

El reconocimiento de números escritos a mano es considerado como el “*Hola Mundo!*” del Deep Learning. Este primer ejemplo fue el que hizo que todo el mundo pusiera la atención en el Deep Learning debido a los excelente resultados que se lograron después de que por muchos años no se lograron avances significativos.

Como en todo proceso que utiliza Redes Neuronales, se requiere un conjunto de datos (dataset) para poder entrenar nuestra red. Para este ejemplo se requiere de un dataset que contenga una gran cantidad de imágenes de números escritos a mano.

En 1998 se creó el “*Modified National Institute of Standards and Technology database*”, mas conocido como MNIST (por sus siglas en inglés), el cual es un dataset que fue creado con la unión de 2 conjuntos de datos (figura 69):

- Imágenes creadas por alumnos de escuelas primarias de Estados Unidos.
- Imágenes creadas por empleados de la Oficina de Censos de Estados Unidos.



Figura 69: Ejemplos del dataset MNIST

El dataset original contiene 60,000 muestras para el entrenamiento y 10,000 muestras de prueba y todas sus imágenes fueron digitalizadas y normalizadas en cuanto a formato de imagen y en tamaño de cada imagen para poderse incorporar al proceso de entrenamiento.



Material Adicional: Se puede consultar el sitio original donde todavía se encuentra el dataset de MNIST original: <http://yann.lecun.com/exdb/mnist/>

Para nuestro ejemplo utilizaremos la biblioteca Scikit-Learn en Python, el cual incluye un subconjunto del "ML hand-written digits dataset" de la Universidad de California en Irvine. Este dataset contiene 5620 muestras (3823 para entrenamiento y 1797 para pruebas) pero el que contiene Scikit-Learn es mas reducido ya que solamente tiene 1797 muestras. En este dataset se almacenan imágenes de los números en forma de matrices de 8x8, cada uno con valores que van de 0 a 16, los cuales representan la intensidad de los pixeles.



Material Adicional: Se puede consultar el sitio original del dataset que utiliza Scikit-Learn: <https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>



Buenas Prácticas: Utilizar siempre los dataset originales para mejores resultados y descargarlos de sus sitios oficiales.

A continuación se trabajará con “*TensorFlow*”, la cual es una biblioteca de código abierta desarrollada inicialmente por Google y que está orientada para trabajar con Machine Learning e Inteligencia Artificial.



Advertencia: Hasta octubre del 2020 TensorFlow sólo se podía ejecutar en Python de 64 bits y de las versiones 3.5 a la 3.8, por lo que si no se cumplen las especificaciones anteriores, no podremos trabajar con TensorFlow.



Buenas Prácticas: Para solucionarlo se puede hacer lo siguiente:

- Trabajar de manera local con contenedores (con las versiones adecuadas) ó trabajar en la nube con alguna herramienta como Google Colaboratory.

Para trabajar con “Google Colaboratory” (figura 70), abrimos su página Web con un navegador (se requiere tener una cuenta de Google):

<https://colab.research.google.com/>



Figura 70: Entorno de Google Colaboratory

Hacemos click en un botón azul en la esquina superior derecha que dice:

Acceder

Nos preguntará el login y password de una cuenta de Gmail, luego seleccionamos “Archivo → Nuevo notebook” y se observará una pantalla blanca para empezar a trabajar: Lo primero es cargar las bibliotecas necesarias:

```
# Bibliotecas necesarias
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.utils import to_categorical
```

El siguiente paso es cargar el dataset de MNIST (el que trae de manera interna Scikit-Learn y que se encuentra en un servidor de Google, esto por estar utilizando Google Colab en línea) e imprimir el valor número 8 del dataset (el cual corresponde en este caso al número “1”):

```
print("\nReconocimiento de números")

# Cargamos dataset
(x_entrenamiento, y_entrenamiento), (x_prueba, y_prueba) =tf.keras.datasets.mnist.load_data()

# Imprimir el dato 8 de entrenamiento
print("\nImpresión de dato de pueba")
plt.imshow(x_entrenamiento[8], cmap=plt.cm.binary)
print("Etiqueta: ", y_entrenamiento[8])
```

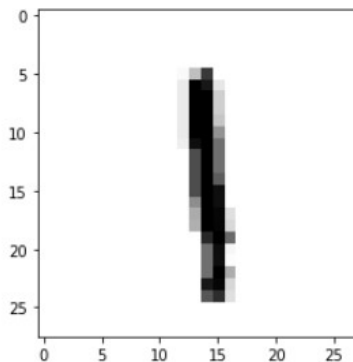
En pantalla se observa el lugar de descarga del dataset y la impresión del número “1” con su respectiva etiqueta “1” (recordar que es un “*Aprendizaje Supervisado*” y que por lo tanto cada imagen gráfica se le asigna una etiqueta que lo representa, así que no se debe confundir la imagen con su etiqueta):

```
Reconocimiento de números
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [=====] - 0s 0us/step
11501568/11490434 [=====] - 0s 0us/step
```

Reconocimiento de números

Impresión de dato de pueba

Etiqueta: 1



Para comprobar las características del dataset, ahora imprimimos las dimensiones del “tensor” que estamos utilizando (se le llama “*tensor*” a la estructura de datos usada en sistemas de Machine Learning), así como el contenido de cada dimensión, es decir, nuestro dataset está definido como un “Tensor”:

```
# Mostrar dimensiones de nuestro tensor
print("Dimensiones del tensor de entrenamiento: ", x_entrenamiento.ndim)

# Mostrar el contenido del tensor
print("Contenido del Tensor: ", x_entrenamiento.shape)
```

En pantalla observaremos que nuestro “*tensor*” y sus dimensiones, en este caso son 3, es decir, lo podemos visualizar como un arreglo de 3 dimensiones. Después al imprimir el contenido de cada dimensión podemos observar lo siguiente en cada dimensión:

- Dimensión 1 = 60000 (el conjunto de imágenes que se van a utilizar para el entrenamiento).
- Dimensión 2 = 28 (el tamaño de renglones que ocupa cada una de las imágenes).
- Dimensión 3 = 28 (el tamaño de columnas que ocupa cada una de las imágenes).

```
Dimensiones del tensor de entrenamiento:  3
Contenido del Tensor:  (60000, 28, 28)
```

Ahora para asegurarnos de no tener problemas en el formato de los datos de entrenamiento que se introducirán en nuestra red neuronal, procederemos a hacer un pre-procesamiento de los datos, convirtiendo todo el dataset de entrenamiento a formato “*float32*”:

```
# Convertir todos los datos a float32 para evitar problemas de formato
x_entrenamiento = x_entrenamiento.astype('float32')
y_entrenamiento = y_entrenamiento.astype('float32')
```

De la parte anterior no saldrá nada en pantalla a no ser que haya algún error. El siguiente paso es asegurarnos de que nuestros datos de entrenamiento “*sólo contengan valores entre 0 y 255*” (ya que es una imagen en escala de grises), por lo tanto se hará un re-escalado de los valores que contengan el dataset para que solamente tengan valores de 0 a 255, y esto se hace de la siguiente manera:

```
# Escalar los datos solamente para un rango de 0 a 255
x_entrenamiento /= 255
y_entrenamiento /= 255
```

Ahora vamos a convertir nuestra matriz (la de entrenamiento y la de prueba) a un vector, para poder alimentar de datos a nuestra red neuronal (esto se hace ya que nuestra entrada es una imagen en 2 dimensiones de 28x28 y la entrada a la red neuronal es solamente en una dimensión, por lo cual hay que pasarla de 28x28 a un vector de 784):

```
# Convertir la matriz de entrenamiento y la de prueba de 28x28 a un solo vector de 784
x_entrenamiento = x_entrenamiento.reshape(60000, 784)
x_prueba = x_prueba.reshape(10000, 784)
```

El siguiente paso es convertir las etiquetas de entrenamiento (por ejemplo una imagen que tiene un 7, su etiqueta es "7") a un formato de salida para nuestra red:

```
# Convertir las etiquetas de datos de un número a una salida de 10 números
# Ejemplo: La etiqueta "7" se convierte a [0,0,0,0,0,0,0,1,0,0]
# Esto es necesario para alimentar la red neuronal
y_entrenamiento = to_categorical(y_entrenamiento, num_classes=10)
y_prueba = to_categorical(y_prueba, num_classes=10)
```

Hasta este punto se cargaron los datos y se les aplicó un pre-procesamiento para poder aplicarlos a nuestro modelo de red neuronal, el siguiente paso es definir la red que se va a utilizar. Dentro de Keras existen 3 modelos de redes neuronales:

- Sequential: Ideal para capas con una entrada y una salida por cada una de ellas.
- Functional API: Para redes flexibles, con múltiples entradas y múltiples salidas.
- Subclassing: Para desarrollar e investigar nuevas formas de conectar redes.



Material Adicional: Para conocer a detalle los modelos de redes neuronales que utiliza Keras se puede consultar el sitio oficial: <https://keras.io/api/models/>

De las clases anteriores seleccionamos "*Sequential*" ya que es la que mas se adecúa para nuestro propósito de reconocer números, por lo que para seleccionar este modelo agregamos a nuestro código lo siguiente:

```
# Seleccionar el modelo de red neuronal a generar
modelo = tf.keras.Sequential()
```

Lo siguiente es crear las capas de nuestra red neuronal y seleccionar la "*Función de Activación*" por cada una de las capas que se vayan agregando, que puede ser la misma para todas las capas, ó cada capa puede tener una diferente función de activación. Algunos puntos importantes sobre las "*Funciones de Activación*" son las siguientes:

- Las funciones que gobiernan el comportamiento de la neurona artificial se llaman funciones de activación.
- La transmisión de esa entrada se conoce como forward propagation.
- Las funciones de activación transforman la combinación de entradas, pesos y sesgos.
- Los productos de estas transformaciones se ingresan para la siguiente capa de nodo.
- Muchas (aunque no todas) las transformaciones no lineales usadas en redes neuronales transforman los datos a rango conveniente, por ejemplo [0,1] ó [-1,1].
- Cuando una neurona artificial pasa de un valor distinto de cero a otro, decimos que esa neurona se ha activado.

Algunas de las funciones de activación más conocidas usadas para activar redes neuronales son las siguientes:

- Sigmoide.
- Softmax.
- Unidad Rectificada Lineal ó Rectified Linear Unit ("ReLU", por sus siglas en inglés).
- Unidad lineal de error gaussiano ó Gaussian Error Linear Unit ("GELU", por sus siglas en inglés).

La función de activación “*Sigmoide*” (figura 71) es de las mas utilizadas, ya que sirve para modelar muchos de los procesos naturales, en donde existe una progresión temporal, desde unos niveles bajos hasta unos niveles altos. En su parte central se tiene una fuerte aceleración donde se ejecuta el cambio de nivel, de alto a bajo, tal como se observa en la siguiente figura:

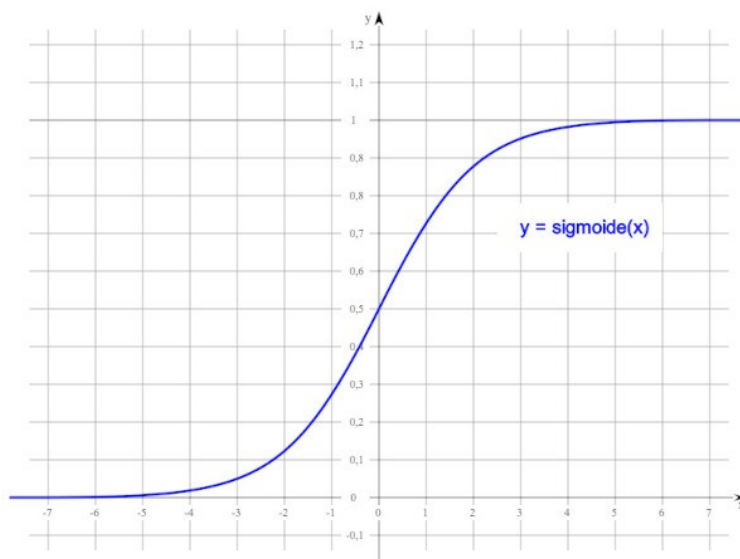


Figura 71: Función de Activación Sigmoide

La función de activación “*Softmax*” calcula las probabilidad de cada clase sobre todas las clases posibles por lo tanto es ideal para utilizar en problemas de clasificación multiclase. Este tipo de función de activación es muy utilizada en la capa final de un modelo de red neuronal ya que nos arroja como salida un rango de probabilidad entre 0 y 1, y la suma de todas las probabilidades de salida nos dará 1.

La función de activación “*ReLU*” (figura 72) es básicamente una función de tipo rampa, en donde con una entrada negativa no tenemos salida, y la salida empieza a crecer en la misma proporción que la entrada a partir de valores positivos de entrada (como se puede observar en la siguiente figura). Este tipo de función se empezó a aplicar a los sistemas de tipo Deep Learning por ofrecer mejores resultados que cuando se aplicaban las funciones de tipo Sigmoide.

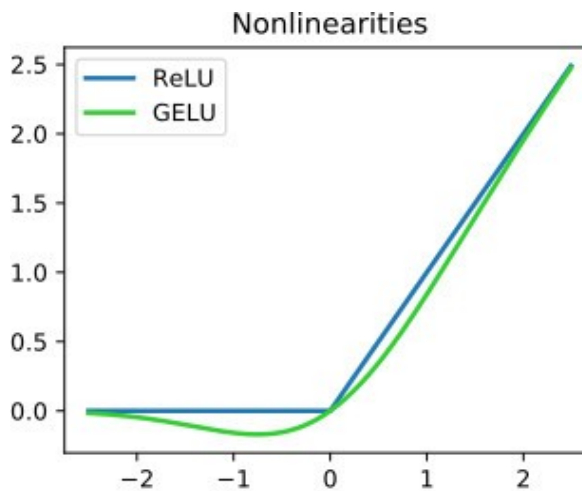


Figura 72: Función de Activación ReLU y GELU

La función de activación “*GELU*” (figura 72) es muy parecida a la ReLU, con la diferencia de que cuando la entrada es negativa y se aproxima a cero, se tiene un pequeño brinco antes de llegar a cero y continúa hasta aumentar de manera positiva (como se observa en la figura anterior). Con esto se logra una transición mas suave teniendo mejores resultados, por lo que es utilizada en los sistemas mas nuevos que han surgido.

La biblioteca Keras soporta las siguientes funciones de activación

- ReLU
- Softmax
- LeakyReLU
- PReLU
- ELU
- ThresholdedReLU



Material Adicional: Para conocer mas sobre las “*Funciones de Activación*” que soporta Keras se puede consultar su sitio oficial: <https://keras.io/api/layers/activations/>

De acuerdo a las “*Funciones de Activación*” anteriores, para nuestro ejemplo seleccionaremos una “*Sigmoide*”, por lo que para crear la primera capa de nuestra red neuronal con 784 entradas (que es la multiplicación de 28x28 que es el tamaño de cada imagen) agregamos el siguiente código:

```
# Crear la primer capa de la red neuronal con activación "sigmoide"
modelo.add(tf.keras.layers.Dense(10, activation='sigmoid', input_shape=(784,)))
```

Ahora agregamos a nuestro modelo de red neuronal una segunda capa con una función de activación “*Softmax*”:

```
# Crear la segunda capa de la red neuronal con activación "softmax"
modelo.add(tf.keras.layers.Dense(10, activation='softmax'))
```

Para ir comprobando el modelo de red neuronal que se está proponiendo, se va a imprimir un resumen de las 2 capas propuestas, para ello se agrega lo siguiente:

```
# Imprimir un resumen del modelo de red neuronal propuesto
print("Modelo Propuesto")
print(modelo.summary())
```

Y se observa en pantalla las 2 capas de nuestra red neuronal.

```
Modelo Propuesto
Model: "sequential_10"

Layer (type)                 Output Shape         Param #
-----
dense_18 (Dense)             (None, 10)          7850
dense_19 (Dense)             (None, 10)          110
-----
Total params: 7,960
Trainable params: 7,960
Non-trainable params: 0
```

Antes de proceder al entrenamiento de la red, es necesario especificar algunas características de nuestro modelo como son las siguientes:

- Función de coste.
- Optimizador.
- Métrica.

La “*Función de Coste*” trata de determinar el error entre el valor estimado y el valor real, con el fin de optimizar los parámetros de la red neuronal. Las funciones de coste mas utilizadas son:

- Raíz cuadrada media (RMSE).
- Error absoluto medio (MAE).
- Error absoluto medio escalado (MASE).
- Entropía cruzada categórica (Categorical Cross-Entropy).
- Entropía cruzada binaria (Binary Cross-Entropy).

Las funciones de coste disponibles en Keras son las siguientes:

- BinaryCrossentropy class.
- CategoricalCrossentropy class.
- SparseCategoricalCrossentropy class.
- Poisson class.
- binary_crossentropy function.
- categorical_crossentropy function.
- sparse_categorical_crossentropy function.
- poisson function.
- KLDivergence class.
- kl_divergence function.



Material Adicional: Para conocer mas sobre las “*Funciones de Coste*” que soporta Keras se puede consultar su sitio oficial: <https://keras.io/api/losses/>

Un “*Optimizador*” lo que hace es obtener mejores valores de los parámetros para reducir el error cometido por la red neuronal. El proceso mediante el cual se hace esto se conoce como “*backpropagation*”. Algunos de los optimizadores mas usados son:

- Stochastic Gradient Descent (SGD).
- Adagrad.
- Adadelta.
- RMSprop.
- Adam.
- Nadam.

Los optimizadores disponibles en Keras son los siguientes:

- SGD.
- RMSprop.
- Adam.
- Adadelta.
- Adagrad.
- Adamax.
- Nadam.
- Ftrl.



Material Adicional: Para conocer mas sobre los “*Optimizadores*” que soporta Keras se puede consultar su sitio oficial: <https://keras.io/api/optimizers/>

Las “*Métricas*” son las funciones que se usan para evaluar el desempeño del modelo. Hay varios tipos de métricas y se clasifican de la siguiente manera:

- Exactitud.
- Probabilísticas.
- Regresión.
- Clasificación.

Las métricas de exactitud disponibles en Keras son:

- Accuracy class.
- BinaryAccuracy class.
- CategoricalAccuracy class.
- SparseCategoricalAccuracy class.
- TopKCategoricalAccuracy class.
- SparseTopKCategoricalAccuracy class.



Material Adicional: Para conocer mas sobre las “*Métricas*” que soporta Keras se puede consultar su sitio oficial: <https://keras.io/api/metrics/>

Para este modelo que se está proponiendo asignaremos la función de coste “*categorical_crossentropy*”, el optimizador “*sgd*” y la métrica “*accuracy*” por ser las mas adecuadas para un modelo pequeño de red neuronal, así que agregamos el código necesario para configurar estos parámetros y aplicarlos a nuestro modelo propuesto por medio del siguiente código:

```
# Especificar modelo
# Función de coste: "categorical_crossentropy"
# Optimizador:      "sgd (Stochastic Gradient Descent)"
# Métrica:          "accuracy"
modelo.compile(loss="categorical_crossentropy", optimizer="sgd", metrics=["accuracy"])
```

Con todos los pasos anteriores ya tenemos definida completamente nuestro modelo de red neuronal, ahora tenemos que pasar a la segunda etapa, la de “*entrenamiento*”, por lo que tendremos que asignar cuáles serán nuestros datos de entrenamiento y la cantidad de iteraciones que hará nuestro modelo (más conocido como “*Épocas*”), que para este caso serán 5 (para no tardar en mucho tiempo de cómputo), así que se agrega lo siguiente:

```
# Entrenamiento del modelo para 5 épocas
modelo.fit(x_entrenamiento, y_entrenamiento, epochs=5)
```

En pantalla podemos observar las 5 *Épocas* que se estuvo entrenando la red neuronal así como su respectiva exactitud:

```
Epoch 1/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.1053 - accuracy: 0.9857
Epoch 2/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.0058 - accuracy: 1.0000
Epoch 3/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.0032 - accuracy: 1.0000
Epoch 4/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.0022 - accuracy: 1.0000
Epoch 5/5
1875/1875 [=====] - 3s 2ms/step - loss: 0.0017 - accuracy: 1.0000
```

Ahora viene la tercera parte donde ya se probará el modelo con los datos de prueba, por lo que agregamos lo siguiente:

```
# Probar modelo con datos de prueba
prueba_perdida, prueba_precision = modelo.evaluate(x_prueba, y_prueba)

# Imprimir precisión
print("Precisión: ", prueba_precision)
```

Y en pantalla observamos lo siguiente:

```
Precisión: 0.09799999743700027
```

Lo cual no indica un nivel de precisión de 97.99% el cual ya es un valor muy aceptable de comportamiento del modelo. El último paso es entonces poder predecir un nuevo número y para este caso se le asignará el número en la posición número "11" (no confundir con el número 11), aunque debería poder predecir otros números que estén fuera del dataset, siempre y cuando cumplan con las restricciones de los datos (tamaño y formato), por lo que agregamos lo siguiente:

```
# Predecir el número que se encuentra en la imagen "11"
prediccion = modelo.predict(x_prueba)
np.argmax(prediccion[11])
print("Predicción: ", prediccion[11])
```

En pantalla aparece el vector de salida que corresponde a la probabilidad de cada uno de los posibles números a predecir (ordenados del 0 al 9):

```
Predicción: [9.9946100e-01 6.2984916e-05 4.0244464e-05 8.3935483e-05 2.2528624e-05
5.7418518e-05 8.1070371e-05 4.5431490e-05 5.6502726e-05 8.9010871e-05]
```

El modelo predice que es el número "0", ya que, en los valores de salida de la red neuronal, el número más alto es el primero

Si el resultado no es bueno, se pueden hacer lo siguientes ajustes:

- Entrenar más veces nuestra red neuronal (asignar más "épocas") para mejorar los valores de predicción.
- Cambiar alguno de estos parámetros:
 - "función de coste".
 - "optimizador".
 - "métrica".
- Agregar más imágenes al entrenamiento.

Capítulo

Procesamiento del Lenguaje Natural

8

Objetivo: Conocer la forma en que una máquina puede interactuar a través del lenguaje natural

“Universos virtuales de complejidad ilimitada pueden ser creados en la forma de programas de computadora”

Joseph Weizenbaum

El alemán Joseph Weizenbaum (1923-2008) fue “Profesor Emérito del MIT”. En 1955 trabajó en la primera computadora que se usó para un banco. En 1963 entra al MIT y crea el famoso “Departamento de Ciencias de la Computación”. En 1966 crea un programa llamado “Eliza”, el cual utilizaba procesamiento del lenguaje natural para entablar conversaciones con usuarios, teniendo gran impacto mundial. Este programa es el precursor de los modernos chatbots actuales.



Este capítulo nos lleva a entender la manera en que ha logrado que las máquinas pueden entender mejor la forma de comunicación de los humanos, a lo que se ha denominado el “*Lenguaje Natural*”. Muchos de los procesos actuales que se usan dentro del área de tecnología utilizan de alguna u otra forma el procesamiento del lenguaje, el cual viene dentro de los textos, imágenes, fotos, videos, páginas Web, sonido y un sinfín de formatos donde se puede encontrar y extraer texto para su posterior procesamiento. Por lo anterior se puede entender la importancia de este tema para el avance de la tecnología actual y para el manejo de la gran cantidad de datos que se generan por segundo en todo el mundo y que ya no es posible para un humano el manejo y entendimiento en tiempo real de tan gran cantidad de información, por lo que las máquinas desempeñan un papel muy importante para el procesamiento de esta información.

8.1 Conceptos Básicos

8.1.1 Definición

El “*Procesamiento del Lenguaje Natural*” ó “*NLP*” (“*Natural Language Processing*” por sus siglas en inglés) es una intersección entre las Ciencias de la Computación, la Inteligencia Artificial y la Lingüística. Para abreviar se utilizarán las siglas de NLP para referirse a este tema ya que así es mejor conocido y fácil de ubicar este concepto. El NLP comprende desde entender partes de un texto

(algo relativamente sencillo) hasta tener una conversación con preguntas y respuestas entre una máquina y un humano, lo cual puede ser muy complejo.

8.1.2 Aplicaciones

El NLP es una de las áreas de mayor investigación, desarrollo e impacto en la sociedad, ya que las grandes tecnológicas le están invirtiendo grandes cantidades de dinero para su desarrollo.

Algunas de las aplicaciones en las que se pueden utilizar el NLP son las siguientes, en orden de complejidad, de las mas sencillas a las mas complicadas (Vajjala et al., 2020):

- Corrección ortográfica.
- Búsqueda de palabras clave en textos.
- Clasificación de textos.
- Extracción de información.
- Agentes conversacionales de un solo dominio.
- Resumen de textos.
- Aplicaciones de preguntas y respuestas.
- Traducción de lenguaje.
- Agentes conversaciones de dominio abierto.

8.1.3 Lenguaje Humano

Lingüística es el estudio del “*Lenguaje*”, y el “*Lenguaje*” es un sistema de comunicación estructurado que implica una combinación compleja de componentes (caracteres, palabras, sentencias, etc.). El lenguaje humano está compuesto básicamente de 4 bloques principales:

- Fonemas.
- Morfemas y Lexemas.
- Sintaxis.
- Contexto.

Los **Fonemas** son la unidad mas pequeña de sonido en un lenguaje, y no tienen un significado como tal, hasta que se conjuntan con otras palabras.

Los **Morfemas** son la unidad mas pequeña del lenguaje que están compuestas por Fonemas y que tienen un significado.

Los **Lexemas** son las variaciones estructurales de los Morfemas relacionados unos con otros por un significado.

Sintaxis es el conjunto de reglas para construir sentencias gramaticalmente correctas de un lenguaje.

Contexto es cuando se unen varios componentes de un lenguaje para converger en un significado especial.

Los humanos hemos creado a través de la creatividad de nuestro lenguaje un conjunto de ideas que pueden ser muy ambiguas para ser procesadas por una máquina, es por eso que es todo un desafío entender correctamente el lenguaje para poder tener una conversación entre una máquina y un humano. La complejidad del lenguaje va asociado a la ambigüedad del lenguaje (en especial en el Español y más en México), ya que un texto puede ser interpretado de muchas maneras, incluso un

mismo humano podría no entenderlo o entenderlo diferente a otro. Un ejemplo es el micro relato “El Dinosaurio”, del escritor Guatemalteco (de origen Hondureño) Augusto Monterroso, que dice así:

“Cuando despertó, el dinosaurio todavía estaba allí...”

Se pueden construir cientos de escenarios, ideas y presunciones de lo que ocurrió en este micro relato (y cada vez que lo leo se me ocurre uno más), y el lector de este libro puede pensar e inventar muchos más. Y viene la pregunta difícil de contestar ¿qué pensaría una máquina acerca de este micro relato, si los mismos humanos no lo pueden entender?

El objetivo de este capítulo precisamente es entender la forma en que las máquinas pueden entender el lenguaje humano, y no sólo de entenderlo, si no también de encontrar la manera de interactuar entre humanos y máquinas.

8.1.4 Historia

Cuando Alan Turing (figura 73) publicó en 1950 su artículo "*Computing Machinery and Intelligence*", donde propuso el famoso "*Test de Turing*" (llamado posteriormente así en su nombre) abrió las puertas para la creación del área de "*Inteligencia Artificial*" y de entrada también la del "*Procesamiento del Lenguaje Natural*" ya que el "*Test de Turing*" incluye el procesamiento e interpretación automática del lenguaje.

Posterior a Turing, el investigador norteamericano Noam Chomsky (figura 74) desarrolló varias teorías derivadas de sus investigaciones sobre el lenguaje, como fueron sus trabajos denominados "*Estructuras Sintácticas*", la "*Teoría de la Gramática Universal*", la "*Gramática Generativa*", la "*Jerarquía de Chomsky*" y la "*Programación Minimalista*". Por todos estos trabajos se le denominó el "*Padre de la Lingüística Moderna*". Sus trabajos se siguen aplicando y utilizando dentro del área de NLP y en muchas otras áreas del conocimiento.

A partir de la década de 1950 surgieron varias corrientes de pensamiento, de las cuales varios investigadores coinciden en 3 etapas importantes, tal como las define Ankur Patel (Patel et al., 2021) se pueden enumerar como las siguientes:

- Basados en Reglas.
- Clásica ó Tradicional.
- Basada en Redes Neuronales

La era basada en "*Reglas*" (o del "*Simbolismo*" como le llaman otros autores) es aquella en donde los investigadores utilizaron un conjunto de reglas donde estructuraban el lenguaje natural, y a partir de ahí, se podían encontrar relaciones y nuevos conocimientos. Esta época abarcó aproximadamente de los años 1950's hasta a principios de los 1990's. En este mismo tiempo es cuando a partir de un conjunto de reglas se crea uno de los primeros sistemas de interacción "*Hombre-Máquina*" de manera escrita que tuvo gran impacto a nivel mundial. Este sistema es "*Eliza*" y fue desarrollado por el investigador alemán Joseph Weizenbaum (figura 42) en 1966.

La era "*Clásica*" o "*Tradicional*" (o de los "*Estadísticos*" como le llaman otros autores) abarcó aproximadamente de 1990 al 2010 y se caracterizó por el aumento en el poder de cómputo y también en el incremento en la cantidad de datos, además de la aplicación de algoritmos que utilizan Probabilidad, Estadística y el naciente "*Machine Learning*". En esta etapa se lograron los primeros sistemas de traducción automática como el que hizo IBM para el gobierno de Canadá para la

traducción automática de documentos gubernamentales. También en esta época surgió la Web, lo que permitió tener gran cantidad de páginas y sitios para poder analizar.

La época de las "Redes Neuronales" (o del "Deep Learning" como le dicen otros autores) abarca aproximadamente desde el año 2010 hasta la actualidad, y como su nombre lo dice, se caracteriza por la aplicación de técnicas de redes neuronales, especialmente de "Deep Learning" al NLP. Usando esta técnica se han logrado grandes avances en poco tiempo, además de crecer de manera exponencial la cantidad de datos que se están generando actualmente lo que ayuda a tener mejores resultados.



Figura 73: Alan Turing

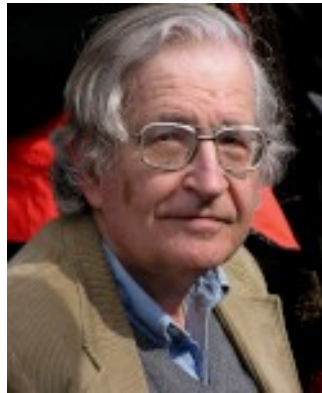


Figura 74: Noam Chomsky



Figura 75: Joseph Weizenbaum

8.1.5 Herramientas de Desarrollo

Algunas de las bibliotecas más utilizadas para el NLP son las siguientes (ordenadas de manera cronológica):

- NLTK (2001)
- spaCy (2015)
- Pytorch (2016)

El "Kit de Herramientas para Lenguaje Natural" ó "Natural Language Toolkit", mas conocido como NLTK (por sus siglas en inglés), es un conjunto de bibliotecas escritas en el año 2001 para el lenguaje Python que fueron desarrolladas por Steven Bird y Edward Loper de la Universidad de Pennsylvania. NLTK es una excelente herramienta para procesar NLP ya que contiene ya datos de prueba y muchas funciones para poder iniciar el procesamiento de textos.

spaCy es un biblioteca desarrollada por Matthew Honnibal e Ines Montani en 2015 y fue escrita para los lenguajes Python y Cython. Por ser una biblioteca mas moderna ya tiene mayor capacidad de procesamiento y soporte para funciones de Deep Learning y puede procesar en mas de 17 lenguajes y segmentación (tokenización) en 65 lenguajes.

Pytorch es una biblioteca que se publicó en 2016 y que inicialmente fue desarrollada por Meta AI (el laboratorio de investigación de Facebook) basado en el lenguaje Torch. Fue orientado para Inteligencia Artificial, principalmente para el procesamiento de imágenes y de NLP. Se basa en "Tensores", que es una forma de vectores para trabajar con modelos de Deep Learning y que tienen soporte para procesarse en GPU's para lograr procesamientos más rápidos.

8.2 Análisis de Textos

8.2.1 Corpus y Fuentes Léxicas

En las técnicas de análisis de datos que se han visto durante los capítulos anteriores de este libro siempre se inicia con la búsqueda de los datos, los cuales siempre surgen de una fuente específica, como puede ser un archivo almacenado en una computadora, un página Web ó un dataset. En el caso del NLP su base de análisis son los textos que se agrupan en denominados "Corpus" (en el área de la Lingüística se le denominan "Corpus Lingüísticos").

Normalmente un "Corpus" es una archivo de texto estructurado (o conjunto de archivos, aunque en plural se denominarían "Corpora"), que contiene un texto en un solo idioma. El contenido de un "Corpus" puede ser un discurso, una noticia, un Tweet, un correo electrónico, etc..

Algunos "Corpus" pueden incorporar en el mismo archivo (o en un archivo separado) los metadatos, como pueden ser:

- Fecha.
- Categoría.
- Autor.
- Idioma.

8.2.2 Análisis Estadístico

Los primeros análisis muy básicos que se pueden hacer a algún texto pueden ser:

- Cantidad de caracteres.
- Cantidad de palabras.
- Palabras más repetidas.
- Distancia entre las palabras más repetidas.
- Distribución de las palabras más repetidas durante todo un texto.

Para iniciar a trabajar de manera práctica se requiere instalar NLTK.



Material Adicional: Para conocer la instalación de NLTK se puede consultar su sitio oficial: <https://www.nltk.org/install.html>

El primer ejemplo práctico será el siguiente:



Actividad: Determinar la "Diversidad Léxica" de un texto:

La "Diversidad Léxica" es un parámetro que nos indica la "riqueza" de un texto, es decir, la forma de expresión de un texto, donde por lo general, una mayor cantidad de palabras utilizadas en un mismo texto, nos lleva a considerar que tiene mas "riqueza" léxica. Hay varias formas de calcular la "Diversidad Léxica" (la cual ya lleva muchos años siendo estudiada), pero para este ejemplo utilizaremos una fórmula muy sencilla:

- Diversidad Léxica = cantidad de palabras / cantidad de caracteres

Para iniciar necesitamos importar NLTK:

```
# Importar bibliotecas
import nltk # Procesamiento del lenguaje natural
```

El siguiente paso es definir un texto a analizar, para esto caso se seleccionará un párrafo de la poesía “*En Paz*” del autor Amado Nervo:

```
# Definir una frase
frase_amado_nervo = "Amé, fui amado, el sol acarició mi faz. ¡Vida, nada me debes! ¡Vida, estamos en paz!"

# Definir la frase a procesar
texto = frase_amado_nervo

# Impresión en pantalla de la frase seleccionada
print("\nTexto Completo:\n", texto)
```

En pantalla se observa lo siguiente:

```
Texto Completo:
Amé, fui amado, el sol acarició mi faz. ¡Vida, nada me debes! ¡Vida, estamos en paz!
```

Lo siguiente es calcular la cantidad de caracteres y la cantidad de palabras que contiene nuestro texto, para lo cual agregamos el siguiente código:

```
# Cantidad de caracteres
caracteres = len(texto)
print("\nCantidad de caracteres: ", caracteres)

# Cantidad de palabras
palabras = len(set(texto))
print("Cantidad de palabras: ", palabras)
```

En pantalla se observará el resultado:

```
Cantidad de caracteres: 84
Cantidad de palabras: 26
```

Ya solo nos falta calcular la “*Diversidad Léxica*”:

```
# Diversidad Léxica
diversidad = palabras / caracteres
print("\nDiversidad Léxica: ", diversidad, "\n")
```

Y en pantalla obtendremos el resultado final:

```
Diversidad Léxica: 0.30952380952380953
```

Entre mayor sea este número mayor será la expresividad de nuestro texto a nivel lingüístico.



Advertencia: En el análisis del texto anterior, la cantidad de palabras obtenidas (26) no concuerda con la real (16), debido a que algunos símbolos ó espacios pueden ser considerados también como palabras por la biblioteca utilizada.



Buenas Prácticas: Para obtener un mejor resultado al analizar textos, se recomienda segmentar (“tokenizar”) de manera específica el texto antes de proceder a su análisis.

A continuación se explicará mas a detalle el proceso de segmentación.

8.2.3 Limpieza de Textos

Uno de los primeros pasos para empezar con el análisis del contenido de un archivo es proceder a la limpieza del texto, quitando símbolos y caracteres que por el momento no nos aporten información (aunque para otros tipos de análisis podría ser importante por ejemplo los símbolos de admiración ó interrogación). Hay varias formas posibles de quitar los caracteres no deseados como son los siguientes:

- Revisar el texto caracter por caracter.
- Revisar el texto a través de Expresiones Regulares.
- Revisar el texto utilizando alguna biblioteca especializada.



Actividad Práctica: Eliminar los símbolos y caracteres no deseados de un texto:

Para resolver esto utilizaremos solamente el lenguaje Python y como primer paso definiremos un texto a analizar, para esto caso se seleccionará un párrafo de la poesía “*Hombres necios que acusáis*” (de dominio público) de la autora Sor Juana Inés de la Cruz:

```
# Definir una frase
frase_sor_juana = "Hombres necios que acusáis a la mujer sin razón sin ver que sois la ocasión de lo mismo que culpáis: si con ansia sin igual solicitáis su desdén ¿por qué queréis que obren bien si las incitáis al mal?"

# Definir la frase a procesar
texto = frase_sor_juana

# Impresión en pantalla de la frase seleccionada
print("\nTexto Completo:\n", texto)
```

En pantalla se observa lo siguiente:

```
Texto Completo:
Hombres necios que acusáis a la mujer sin razón sin ver que sois la ocasión de lo mismo que culpáis: si con ansia sin igual solicitáis su desdén ¿por qué queréis que obr en bien si las incitáis al mal?
```

Ahora se hará el procesamiento, que no es mas que hacer un ciclo para recorrer cada carácter del texto y compararlo con cada uno de los caracteres que queremos eliminar (almacenados en la variable “eliminar”, y donde debemos de agregar todos los caracteres indeseables), creando un nuevo texto ya limpio:

```
# Eliminar simbolos y caracteres especiales
eliminar = '()[],. ""??:;_ -!' # Agregar los que falten
texto_sin_simbolos = ""
for caracter in texto:
    if(caracter not in eliminar):
        texto_sin_simbolos = texto_sin_simbolos + caracter
```

Ya solo falta imprimir el nuevo texto ya procesado:

```
# Impresión del texto final sin caracteres
print('\nTexto final sin simbolos ni caracteres especiales:\n', texto_sin_simbolos, "\n")
```

Y pantalla obtendremos el resultado final:

```
Texto final sin simbolos ni caracteres especiales:
Hombres necios que acusáis a la mujer sin razón sin ver que sois la ocasión de lo mi
smo que culpáis si con ansia sin igual solicitáis su desdén por qué queréis que obren
bien si las incitáis al mal
```

Otra técnica es utilizar “*Expresiones Regulares*”, la cual nos permite identificar rápidamente caracteres y patrones de caracteres que se encuentran en un texto y esto lo hace mucho más fácil. Python cuenta de manera interna con una biblioteca para “*Expresiones Regulares*” por lo cual no es necesario instalar ningún componente adicional, sólo importarlo en nuestro código como cualquier biblioteca de la siguiente manera:

```
# Bibliotecas a utilizar
import re # Manejo de expresiones regulares
```

De ejemplo se utilizará el texto anterior, al cual le aplicaremos las expresiones regulares para encontrar y eliminar los caracteres indeseados. El identificador “w” de la expresión regular utilizada nos representa todos los caracteres que no sean números y/o letras (mayúsculas y minúsculas) y el identificador “s” nos representa los espacios en blanco repetidos:

```
# Eliminar simbolos y caracteres especiales usando expresiones regulares
texto_sin_simbolos = re.sub(r'^\w\s','', texto)
```

Ya solo falta imprimir el nuevo texto ya procesado:

```
# Impresión del texto final sin caracteres
print('\nTexto final sin simbolos ni caracteres especiales:\n', texto_sin_simbolos, "\n")
```

Y pantalla obtendremos el resultado final:

```
Texto final sin simbolos ni caracteres especiales:
Hombres necios que acusáis a la mujer sin razón sin ver que sois la ocasión de lo mismo que culpáis si co
n ansia sin igual solicitáis su desdén por qué queréis que obren bien si las incitáis al mal
```



Material Adicional: Para conocer sobre “*Expresiones Regulares*” consultar la documentación oficial de Python: <https://docs.python.org/3/library/re.html>

Otra técnica es utilizando alguna biblioteca especial para el procesamiento del texto y estas se utilizarán para el “*tokenizado*”, el cual se comentará a continuación.

8.2.4 Segmentación de Textos

Una vez que se ha hecho un pre-procesamiento del texto para poder eliminar algunos símbolos y caracteres no deseados, el siguiente paso es la segmentación del texto, es decir, separar el texto en

palabras, donde cada palabra encontrada dentro del texto lo convertiremos a una lista de palabras importantes ó “tokens” (como se le denomina en inglés). Al proceso completo de pasar un texto a un conjunto de palabras importantes para nuestro análisis se le denomina “Tokenización” (que proviene de la palabra “Token” en inglés, y es mas conocido de esta manera).

Para lograr encontrar palabras relevantes dentro de un texto y hacer la “tokenización”, se requiere eliminar las palabra que aporten poca relevancia a nuestro análisis (como son los artículos “el, la, los”, etc.) y a los cuales se les denominó “Stopwords” ó “palabras vacías” en español. Cada lenguaje tiene su propio conjunto de “stopwords” y se deberá tener una lista con las “stopwords” que se vayan a eliminar. En el caso de la biblioteca NLTK, ya se cuenta con esta lista, la cual está disponible para diferentes lenguajes.



Actividad Práctica: Buscar las palabras relevantes de un texto (tokenizar) quitando las “stopwords”.

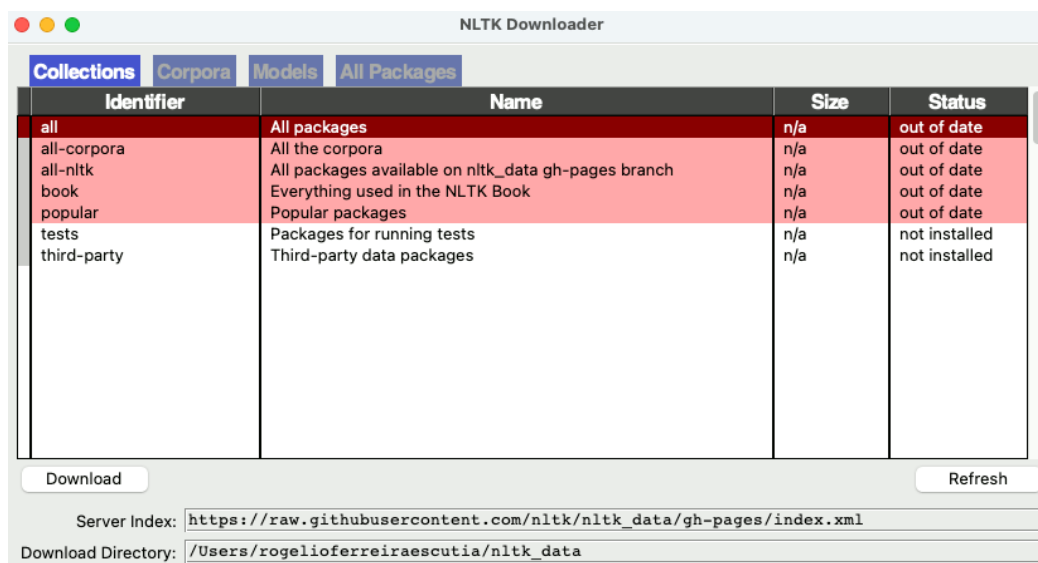
Lo primero será instalar nuestra lista de “stopwords” para utilizarse dentro de NLTK, para ello entraremos a nuestra línea de comandos (terminal) y escribiremos “python3” (ó solamente “python, de acuerdo al que se haya instalado) para invocar al intérprete de Python y se observará lo siguiente:

```
[rogelioferreiraescutia@Mac-mini-de-Rogelio ~ % python3
Python 3.10.0 (v3.10.0:b494f5935c, Oct 4 2021, 14:59:19) [Clang 12.0.5 (clang-1
205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> ]
```

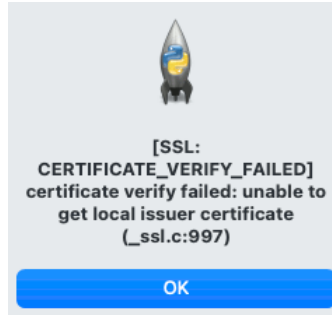
Ya dentro del intérprete vamos a importar NLTK y cargaremos el “NLTK Downloader”, que es la herramienta gráfica para mostrar los “Corpus” y herramientas que se han descargado a nuestra computadora, y para ello escribimos lo siguiente en el intérprete:

```
[>>> import nltk
[>>> nltk.download()
```

Se abre una ventana modo gráfico y se observará la pantalla del “NLTK Downloader”, donde se encuentran todas las herramientas del NLTK que se han descargado de manera local a nuestra computadora:



Algunos sistemas pueden marcar el siguiente error:



Esto es debido a un problema de comunicación del SSL que requiere los certificados locales. En caso de los sistemas con MacOS hay que cambiarse al directorio de Python con la siguiente instrucción en la terminal:

- `cd /Applications/Python\ 3.10`

Y luego instalar los certificados con el siguiente comando:

- `./Install\ Certificates.command`

Se observará en pantalla la instalación:

```
rogelioferreiraescutia@Mac-mini-de-Rogelio Python 3.10 % ./Install\ Certificates
.command
-- pip install --upgrade certifi
Requirement already satisfied: certifi in /Library/Frameworks/Python.framework/V
ersions/3.10/lib/python3.10/site-packages (2021.10.8)
Collecting certifi
  Downloading certifi-2022.6.15-py3-none-any.whl (160 kB)
----- 160.2/160.2 kB 1.2 MB/s eta 0:00:00
Installing collected packages: certifi
  Attempting uninstall: certifi
    Found existing installation: certifi 2021.10.8
    Uninstalling certifi-2021.10.8:
      Successfully uninstalled certifi-2021.10.8
  Successfully installed certifi-2022.6.15
  -- removing any existing file or link
  -- creating symlink to certifi certificate bundle
  -- setting permissions
  -- update complete
```

Nuevamente se repiten los pasos para entrar al intérprete y ya se podrá observar la pantalla del “NLTK Downloader”. Una vez que ya podamos entrar y salir de nuestra pantalla de control de descargas del NLTK (el “NLTK Downloader”), procederemos a la instalación de las “stopwords”. Por medio de la terminal nuevamente invocamos a nuestro intérprete de Python:

- `python3`

Ya dentro del intérprete importamos NLTK y luego importamos las “stopwords” por medio de los siguientes instrucciones:

- `import nltk`
- `nltk.download('stopwords')`

Y observaremos en pantalla que descarga el archivo y lo almacena de manera local:

```
>>> import nltk
>>> nltk.download('stopwords')
[nltk_data] Downloading package stopwords to
[nltk_data] /Users/rogelioferreiraescutia/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
True_
```

Para salir en cualquier momento del intérprete de Python tecleamos lo siguiente:

- `exit()`

Si queremos saber cuáles son las “*stopwords*” que utiliza NLTK para el idioma español, abrimos con un editor de texto el archivo “*spanish*” que se encuentra en el siguiente directorio:

- `/Users/rogelioferreiraescutia/nltk_data/corpora/stopwords/`

Donde el directorio “*rogelioferreiraescutia*” cambia de acuerdo al nombre del usuario que está usando la computadora. O podemos imprimirlo directamente en pantalla con la siguiente instrucción:

- `cat /Users/rogelioferreiraescutia/nltk_data/corpora/stopwords/spanish`

Actualmente se tienen 313 “*stopwords*” consideradas para el idioma español. A partir de este paso ya estamos listos para “*tokenizar*” y eliminar las “*stopwords*” de un texto. El primer paso como siempre es definir las bibliotecas a usar:

```
# Bibliotecas a utilizar
import re # Manejo de expresiones regulares
import nltk # Para procesar lenguaje natural
from nltk.tokenize import word_tokenize # Para Tokenizar un texto
from nltk.corpus import stopwords # Cargar las "Stopwords" del español
```

El siguiente paso es definir el texto a analizar, para esto ejemplo se seleccionará un párrafo de la poesía “*Como soy reina y fui mendiga, ahora...*” de la autora Gabriela Mistral:

```
# Definir una frase
frase_gabriela_mistral = "Como soy reina y fui mendiga, ahora vivo en puro temblor de que me dejes, y te pregunto, pálida, a cada hora: -¿Estás conmigo aún? ¡Ay, no te alejes!- Quisiera hacer las marchas sonriendo y confiando ahora que has venido; pero hasta en el dormir estoy temiendo y pregunto entre sueños: -¿No te has ido?_"

# Definir la frase a procesar
texto = frase_gabriela_mistral

# Impresión en pantalla de la frase seleccionada
print("\nTexto Completo:\n\n", texto)
```

En pantalla se observa lo siguiente:

Texto Completo:

```
Como soy reina y fui mendiga, ahora vivo en puro temblor de que me dejes, y te pregunto, pálida, a cada hora:
-¿Estás conmigo aún? ¡Ay, no te alejes!- Quisiera hacer las marchas sonriendo y confiando ahora que has venid
o; pero hasta en el dormir estoy temiendo y pregunto entre sueños: -¿No te has ido?-
```

Nuevamente quitamos caracteres y símbolos no deseados usando expresiones regulares:

```
# Eliminar simbolos y caracteres especiales usando expresiones regulares
texto_sin_simbolos = re.sub(r'^\w\s', '', texto)
```

Ahora “*tokenizamos*” el texto, lo imprimimos en pantalla junto con el total de tokens:

```
# Convertimos a tokens todo el texto y lo imprimimos en pantalla
tokens_de_mi_texto = word_tokenize(texto_sin_simbolos)
print('\nImpresión de todos los tokens del texto:\n\n', tokens_de_mi_texto)
print('\n Tokens Totales: ', len(tokens_de_mi_texto))
```

En pantalla aparece lo siguiente:

```
Impresión de todos los tokens del texto:

['Como', 'soy', 'reina', 'y', 'fui', 'mendiga', 'ahora', 'vivo', 'en', 'puro', 'temblor', 'de', 'que', 'me',
'dejes', 'y', 'te', 'pregunto', 'pálida', 'a', 'cada', 'hora', 'Estás', 'conmigo', 'aún', 'Ay', 'no', 'te', 'a
lejes', 'Quisiera', 'hacer', 'las', 'marchas', 'sonriendo', 'y', 'confiando', 'ahora', 'que', 'has', 'venido',
'pero', 'hasta', 'en', 'el', 'dormir', 'estoy', 'temiendo', 'y', 'pregunto', 'entre', 'sueños', 'No', 'te', '
has', 'ido']

Tokens Totales: 55
```

A continuación cargamos las “*stopwords*” del idioma español:

```
# Cargamos las "stopwords" del español (las palabras que no nos aportan información)
palabras_vacias = set(stopwords.words('spanish'))
```

Procederemos al filtrado de tokens, por lo que vamos a recorrer nuestra lista de tokens para encontrar si uno de ellos pertenece a la lista de “*stopwords*”, de ser así, no lo pasamos a nuestra nueva lista, la cual ya será nuestra final final, tokenizada y filtrada:

```
:
# Filtramos los tokens eliminando las "stopwords"
lista_final = []
for palabra in tokens_de_mi_texto:
    if palabra not in palabras_vacias:
        lista_final.append(palabra)
```

Por último imprimimos el resultado:

```
# Impresión final de las palabras relevantes del texto (Tokenizar)
print('\nLista Final eliminando las palabras vacías (no relevantes):\n\n', lista_final)
print("\nTotal de Tokens sin Stopwords: ", len(lista_final),"\n")
```

Y en pantalla observamos el resultado final de la tokenización:

```
Lista Final eliminando las palabras vacías (no relevantes):

['Como', 'reina', 'mendiga', 'ahora', 'vivo', 'puro', 'temblor', 'dejes', 'pregunto', 'pálida', 'cada', 'hora',
', 'Estás', 'conmigo', 'aún', 'Ay', 'alejes', 'Quisiera', 'hacer', 'marchas', 'sonriendo', 'confiando', 'ahora',
', 'venido', 'dormir', 'temiendo', 'pregunto', 'sueños', 'No', 'ido']

Total de Tokens sin Stopwords: 30
```

8.2.5 Palabras Importantes

El primer paso para tratar de encontrar el tema (ó los temas principales) de los que trata un texto es encontrar las “*palabras importantes*” que nos indiquen su contenido. La forma mas sencilla de encontrar esto es buscando las palabras que mas se repiten en un mismo texto (quitando las “*stopwords*”), esto nos da la idea del contenido del mismo (aunque no siempre puede ser así, la misma ambigüedad de las palabras y del texto nos pueden llevar a conclusiones erróneas, pero es un primer paso). Para encontrar las palabras mas importantes se realizará el siguiente ejemplo práctico:



Actividad Práctica: Buscar las palabras mas repetidas de un texto.

El primer paso nuevamente es definir un texto a analizar, para este ejemplo se seleccionó el texto de la traducción y transcripción del discurso que dió Steve Jobs en la Universidad de Stanford el 12 de junio del 2005 (el cual es de dominio público). Este discurso se almacenó en un archivo y se puede descargar en el siguiente enlace:

- http://www.xumarhu.net/discurso_steve_jobs_stanford_12_junio_2005.txt

Del sitio anterior se descarga el archivo y se coloca en el mismo directorio donde haremos nuestro código en Python. Para este ejemplo también se hará la limpieza del texto, la “*tokenización*” y eliminación de “*stopwords*”, por lo cual ocuparemos las siguientes bibliotecas:

```
# Bibliotecas a utilizar
import re # Manejo de expresiones regulares
import nltk # Para procesar lenguaje natural
from nltk.tokenize import word_tokenize # Para Tokenizar un texto
from nltk.corpus import stopwords # Cargar las "Stopwords" del español
from collections import Counter # importamos para poder contar las palabras repetidas
from collections import OrderedDict # importamos para ordenar el conteo de palabras repetidas
```

El siguiente paso es cargar a memoria el archivo de texto del discurso de Steve Jobs que descargamos previamente:

```
# Cargar archivo de texto a memoria
with open('discurso_steve_jobs_stanford_12_junio_2005.txt', 'r') as archivo_en_memoria:
    texto = archivo_en_memoria.read()
```

Eliminamos caracteres y símbolos no deseados y “*tokenizamos*”:

```
# Eliminar simbolos y caracteres especiales usando expresiones regulares
texto_sin_simbolos = re.sub(r'^\w\s', '', texto)

# Convertimos a tokens todo el texto y lo imprimimos en pantalla
tokens_de_mi_texto = word_tokenize(texto_sin_simbolos)
print('\nImpresión de todos los tokens del texto:\n\n', tokens_de_mi_texto)
print('\n Tokens Totales: ', len(tokens_de_mi_texto))
```

Cargamos las “*stopwords*” y filtramos el texto:

```
# Cargamos las "stopwords" del español (las palabras que no nos aportan información)
palabras_vacias = set(stopwords.words('spanish'))

# Filtramos los tokens eliminando las "stopwords"
lista_final = []
for palabra in tokens_de_mi_texto:
    if palabra not in palabras_vacias:
        lista_final.append(palabra)
```

Se puede ir imprimiendo en pantalla el resultado hasta este momento agregando lo siguiente:

```
# Impresión final de las palabras relevantes del texto (Tokenizar)
print('\nLista Final eliminando las palabras vacías (no relevantes):\n\n', lista_final)
print("\nTotal de Tokens sin Stopwords: ", len(lista_final))
```

Se cuentan y ordenan las palabras:

```
# Contar y ordenar las palabras repetidas
contador = Counter(lista_final)
print('\nLista de palabras y cuántas veces se repiten (en orden descendente por cantidad de repeticiones)
:\n', contador)
print('Total:', len(contador))
```

Se imprime cada palabra y su repetición en todo el texto conforme van apareciendo:

```
# Contar las palabras repetidas
contador_ordenado = OrderedDict(contador)
print('\nLista de palabras y cuántas veces se repiten (conforme van apareciendo en el texto):\n',
contador_ordenado)
print('Total:', len(contador))
```

Y por último se buscan las 5 palabras más repetidas:

```
# Extraemos sólo las 5 palabras más repetidas de todo el texto
las_mas_repetidas = OrderedDict(contador.most_common(5))
print('\nImpresión de las 5 palabras mas repetidas de todo el texto:\n', las_mas_repetidas)
print('\nTotal:', len(las_mas_repetidas), "\n")
```

El resultado final en pantalla es el siguiente:

```
Impresión de las 5 palabras mas repetidas de todo el texto:
OrderedDict([('vida', 14), ('Y', 14), ('años', 13), ('No', 10), ('Apple', 9)])

Total: 5
```

De lo anterior se puede inferir que el discurso de Steve Jobs trata sobre la “vida”, los “años” y “Apple”, cosas que coinciden al analizar el texto de manera manual, aunque como se comentó anteriormente, no siempre es así. También se observa la palabra “Y” que no tomó como una “stopword”, aunque manualmente podríamos agregarla si consideramos que no aporta mucha información. También apareció repetida muchas veces la palabra “no”, la cual indica que hay muchas negaciones o frases con esta palabra.

8.2.6 Nubes de Palabras

Una forma más gráfica de representar y analizar es cuando se utilizan las “Nubes de Palabras” (“WordCloud” en inglés). Éstas nos representan de manera gráfica las palabras que más se repiten en un texto y la importancia de acuerdo al tamaño de las letras que se imprimen en pantalla. Por ejemplo, la “Nube de Palabras” del discurso de “Steve Jobs” que se acaba de analizar se encuentra en la figura 76:



Figura 76: Nube de Palabras del discurso de Steve Jobs

Donde se observan de manera mas clara las palabras principales que se manejan durante el texto. El siguiente es un ejemplo para la creación de una “Nube de Palabras”.



Actividad Práctica: Generar una “Nube de Palabras” a partir de una página de Wikipedia.

Lo primero es definir la página de Wikipedia, la cual será la de “*Ciencia de Datos*” (en idioma español). La página es la siguiente:

- https://es.wikipedia.org/wiki/Ciencia_de_datos

La pantalla de la página anterior se observa de la siguiente manera:



Del sitio anterior se descargará el texto acerca de la definición y conceptos sobre “*Ciencia de Datos*” (texto de la parte derecha). Empezaremos por cargar nuestras bibliotecas necesarias:

```
# Bibliotecas a utilizar
import wikipedia
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

# Extraer y procesar de Wikipedia
# Manejo de expresiones regulares
# Para procesar lenguaje natural
# Para Tokenizar un texto
# Cargar las "Stopwords" del español
# Crear nubes de ideas
# Graficar
```

Se seleccionará Wikipedia en Español, se cargará la página seleccionada y se imprimirá en pantalla:

```
# Seleccionar Wikipedia en Español
wikipedia.set_lang("es")

# Extraer información de una página de Wikipedia
texto = wikipedia.summary("Ciencia de Datos")

# Impresión completa del texto
print("\nTexto Completo:\n\n", texto)
```



Material Adicional: La lista de todas los idiomas en los que está disponible Wikipedia se pueden consultar en su sitio oficial:

https://meta.wikimedia.org/wiki/List_of_Wikipedias_by_language_group

Nuevamente eliminamos símbolos y caracteres especiales y “*tokenizamos*” el texto:

```
# Eliminar simbolos y caracteres especiales usando expresiones regulares
texto_sin_simbolos = re.sub(r'^\w\s', '', texto)

# Convertimos a tokens todo el texto y lo imprimimos en pantalla
tokens_de_mi_texto = word_tokenize(texto_sin_simbolos)
print('\nImpresión de todos los tokens del texto:\n\n', tokens_de_mi_texto)
print('\n Tokens Totales: ', len(tokens_de_mi_texto))
```

Cargamos las “*stopwords*” y hacemos el filtrado:

```
# Cargamos las "stopwords" del español (las palabras que no nos aportan información)
palabras_vacias = set(stopwords.words('spanish'))

# Filtramos los tokens eliminando las "stopwords"
lista_final = []
for palabra in tokens_de_mi_texto:
    if palabra not in palabras_vacias:
        lista_final.append(palabra)
```

Imprimimos los “tokens” de nuestro texto:

```
# Impresión final de las palabras relevantes del texto (Tokenizar)
print('\nLista Final eliminando las palabras vacías (no relevantes):\n\n', lista_final)
print("\nTotal de Tokens sin Stopwords: ", len(lista_final))
```

Convertimos toda nuestra lista de “*tokens*” en un solo texto, ya que para generar la nube (con la biblioteca “*WordCloud*”) se le tienen que dar los datos en este formato y lo imprimimos en pantalla:

```
# Convertir la lista de tokens a un solo texto separados por un espacio
texto_final = " ".join(lista_final)

# Imprimir el texto final
print("texto final:\n", texto_final)
```

De acuerdo a los parámetros que nos pide la biblioteca “*WordCloud*”, configuramos la pantalla de salida de nuestra “*Nube de Palabras*” con los siguientes datos:

```
# Formato de la "Nube de Ideas"
nube_de_ideas = WordCloud(
    width = 500,
    height = 500,
    random_state = 1,
    background_color = "salmon",
    colormap= "Pastel1",
    collocations = False,
    stopwords = STOPWORDS,
).generate(texto_final)
```

Y por último configuramos el gráfico y lo generamos:

```
# Graficar (generar la "Nube de Ideas")
plt.imshow(nube_de_ideas, interpolation="bilinear")
plt.axis("off")
plt.show()
```

En la pantalla aparecerá otra nueva ventana con nuestra “Nube de Palabras” final para nuestra página de Wikipedia (figura 77):



Figura 77: Nube de Palabras de una página de Wikipedia

Como se observa, las palabras que mas se repiten son las que se utilizan en muchos de los conceptos sobre “Ciencia de Datos”.

8.2.7 Análisis de Sentimientos

El "Análisis de Sentimientos" es el procedimiento para identificar, extraer y cuantificar los estados afectivos de las personas ó grupos de personas, utilizando el "Procesamiento del Lenguaje Natural", el "Análisis del Texto" y la "Lingüística Computacional". Su rango de aplicación es muy grande ya que se puede aplicar para determinar la situación emocional de los clientes en cualquier proceso de servicios y ventas así como el de clasificar la intención de cualquier texto.

Algunas de las aplicaciones actuales mas usadas son:

- Análisis de las experiencias de usuarios con un producto o servicio.
- Análisis de discursos políticos.
- Análisis de artículos y libros.

La forma más fácil de hacer un "Análisis de Sentimientos" es detectando la "polaridad" de una palabra, frase ó texto, y se pueden clasificar en 3:

- Positivo.
- Negativo.
- Neutral.

Existen varias técnicas para hacer estos análisis y cada día surgen mas debido a la necesidad y cantidad de casos donde se puede aplicar. Básicamente podríamos tener la siguiente clasificación de técnicas:

- Manual: Generando listas de palabras y/o frases positivas para identificarlas.
- Automática: Usando bibliotecas que ya incluyen sus algoritmos propios para hacer el análisis.

A continuación vienen algunos ejemplos prácticos para hacer el “*Análisis de Sentimientos*”.



Actividad Práctica: Analizar un texto para determinar su polaridad (positivo, negativo ó neutral).

Lo primero es definir el texto a analizar, para este ejemplo se utilizará el libro completo de “*Don Quijote de la Mancha*” de Miguel de Cervantes Saavedra (de dominio público). Este libro se almacenó en un archivo y se puede descargar de cualquiera de los 2 enlaces siguientes:

- http://www.xumarhu.net/texto_001_don_quijote.txt
- http://sagitario.itmorelia.edu.mx/~rogelio/texto_001_don_quijote.txt

Del sitio anterior se descarga el archivo y se coloca en el mismo directorio donde se colocará el código en Python. Para este ejemplo también se hará la limpieza del texto, la “*tokenización*” y eliminación de “*stopwords*”, por lo cual ocuparemos las siguientes bibliotecas:

```
# Bibliotecas a utilizar
import re # Manejo de expresiones regulares
import nltk # Para procesar lenguaje natural
from nltk.tokenize import word_tokenize # Para Tokenizar un texto
from nltk.corpus import stopwords # Cargar las "Stopwords" del español
```

El siguiente paso es cargar a memoria el archivo de texto del libro de “*Don Quijote de la Mancha*” que descargamos previamente y lo convertimos todo a minúsculas (por los comparaciones que se harán más adelante):

```
# Cargar archivo de texto a memoria
with open('texto_001_don_quijote.txt', 'r') as archivo_en_memoria:
    texto = archivo_en_memoria.read()

# Convertir todo el texto a minúsculas
texto_minusculas = texto.lower()
```

Eliminamos caracteres y símbolos no deseados y “*tokenizamos*”:

```
# Eliminar simbolos y caracteres especiales usando expresiones regulares
texto_sin_simbolos = re.sub(r'[^\w\s]', '', texto_minusculas)

# Convertimos a tokens todo el texto y lo imprimimos en pantalla
tokens_de_mi_texto = word_tokenize(texto_sin_simbolos)
print('\nTokens Totales: ', len(tokens_de_mi_texto))
```

Cargamos las “*stopwords*” y filtramos el texto:

```
# Cargamos las "stopwords" del español (las palabras que no nos aportan información)
palabras_vacias = set(stopwords.words('spanish'))

# Filtramos los tokens eliminando las "stopwords"
lista_final = []
for palabra in tokens_de_mi_texto:
    if palabra not in palabras_vacias:
        lista_final.append(palabra)
```

Imprimimos la cantidad de “tokens” ya sin “stopwords”:

```
# Impresión total de Tokens sin StopWords
print("Total de Tokens sin Stopwords: ", len(lista_final))
```

Hasta este momento se observará en pantalla lo siguiente:

```
Tokens Totales: 187009
Total de Tokens sin Stopwords: 91055
```

Como se observa, al hacer el filtrado de palabras (quitar las “stopwords”) se redujo considerablemente los “tokens” a analizar. Ahora viene la parte del “Análisis de Sentimientos”, y la primer forma de hacerlo es construir 2 listas, una de palabras conocidas comúnmente como positivas y otra para las negativas. En este ejemplo sólo se hicieron las 2 listas con 5 elementos cada una:

```
# Lista de palabras positivas y negativas en español
palabras_positivas = ['bueno', 'encantado', 'bonito', 'increíble', 'superior']
palabras_negativas = ['malo', 'decepcionado', 'feo', 'terrible', 'inferior']
```

Nuestro contador de palabras positivas y negativas lo inicializamos a cero:

```
# Inicializar a cero las cantidades positivas y negativas
numero_de_positivas = numero_de_negativas = 0
```

Ahora buscamos si dentro de nuestro libro hay alguna coincidencia de alguna palabra que está en nuestra lista de palabras positivas y negativas, y si es así, incrementamos su respectivo contador:

```
# Buscamos las palabras positivas y negativas dentro del texto
for elemento in lista_final:
    if elemento in palabras_positivas:
        numero_de_positivas = numero_de_positivas + 1
    if elemento in palabras_negativas:
        numero_de_negativas = numero_de_negativas + 1
```

Ahora ya solo falta imprimir el total de palabras positivas y negativas y la polaridad del texto:

```
# Impresión de resultados
print("\nNúmero de palabras positivas encontradas: ", numero_de_positivas)
print("Número de palabras negativas encontradas: ", numero_de_negativas)
print("\nPolaridad del texto:")
polaridad = numero_de_positivas - numero_de_negativas
if polaridad>0: print("Positiva\n")
elif polaridad<0: print("Negativa\n")
else: print("Neutral\n")
```

En pantalla se observó el resultado final:

```
Número de palabras positivas encontradas: 100
Número de palabras negativas encontradas: 25

Polaridad del texto:
Positiva
```

Se observa que se encontraron 100 palabras positivas y 25 negativas, por lo cual se considera que la polaridad del texto es “*Positiva*”, es decir habla sobre cosas o ideas que se consideran mejores (hablando de palabras utilizadas). Un texto es “*Negativo*” cuando la cantidad de palabras negativas es mayor a las positivas y “*Neutral*” cuando la cantidad de palabras positivas y negativas son iguales.

Hay que recordar que el resultado anterior es basándose solamente en nuestra pequeña lista de 5 palabras positivas y 5 palabras negativas, para obtener un mejor resultado se necesita una lista mas larga de palabras, lo cual se verá en el siguiente ejemplo práctico.



Actividad Práctica: Analiza un texto para determinar su polaridad (positivo, negativo ó neutral) utilizando listas de palabras positivas y negativa

Para este ejemplo utilizaremos el mismo texto anterior (“*Don Quijote de la Mancha*”) para observar y comparar los resultados. Para mejorar el desempeño utilizaremos una lista de palabras “*positivas*” y “*negativas*” mas grande (en lenguaje español), las cuales fueron obtenidas del sitio en GitHub de Jose Cardona Figueroa (de dominio público) y cuya página es la siguiente:

- <https://github.com/JoseCardonaFigueroa/sentiment-analysis-spanish/blob/master/data/subjectivity.csv>

En este sitio se encuentra una lista de palabras “positivas” y “negativas”, y después de un procesamiento se convirtieron a 2 listas por separado en formato txt. La lista “positiva” (que es de 414 palabras) se puede descargar de cualquiera de los siguientes sitios:

- http://www.xumarhu.net/texto_002_palabras_positivas.txt
- http://sagitario.itmorelia.edu.mx/~rogelio/texto_002_palabras_positivas.txt

Y la lista “negativa” (que es de 625 palabras) se puede descargar de cualquiera de los siguientes sitios:

- http://www.xumarhu.net/texto_003_palabras_negativas.txt
- http://sagitario.itmorelia.edu.mx/~rogelio/texto_003_palabras_negativas.txt

Ambos archivos deben descargarse y junto con ella archivo del libro “*Don Quijote de la Mancha*”, deben de colocarse en el mismo directorio donde haremos nuestro código en Python. Para este ejemplo también se hará la limpieza del texto, la “*tokenización*” y eliminación de “*stopwords*”, por lo cual ocuparemos las siguientes bibliotecas:

```
# Bibliotecas a utilizar
import re # Manejo de expresiones regulares
import nltk # Para procesar lenguaje natural
from nltk.tokenize import word_tokenize # Para Tokenizar un texto
from nltk.corpus import stopwords # Cargar las "Stopwords" del español
```

El siguiente paso es cargar a memoria el archivo de texto del libro de “*Don Quijote de la Mancha*” que descargamos previamente y lo convertimos todo a minúsculas (por los comparaciones que se harán más adelante):

```
# Cargar archivo de texto a memoria
with open('texto_001_don_quijote.txt','r') as archivo_en_memoria:
    texto = archivo_en_memoria.read()

# Convertir todo el texto a minúsculas
texto_minusculas = texto.lower()
```

Eliminamos caracteres y símbolos no deseados y “*tokenizamos*”:

```
# Eliminar simbolos y caracteres especiales usando expresiones regulares
texto_sin_simbolos = re.sub(r'^\w\s]', '', texto_minusculas)

# Convertimos a tokens todo el texto y lo imprimimos en pantalla
tokens_de_mi_texto = word_tokenize(texto_sin_simbolos)
print('\nTokens Totales: ', len(tokens_de_mi_texto))
```

Cargamos las “*stopwords*” y filtramos el texto:

```
# Cargamos las "stopwords" del español (las palabras que no nos aportan información)
palabras_vacias = set(stopwords.words('spanish'))

# Filtramos los tokens eliminando las "stopwords"
lista_final = []
for palabra in tokens_de_mi_texto:
    if palabra not in palabras_vacias:
        lista_final.append(palabra)
```

Imprimimos la cantidad de “*tokens*” ya sin “*stopwords*”:

```
# Impresión total de Tokens sin StopWords
print("Total de Tokens sin Stopwords: ", len(lista_final))
```

Hasta este momento se observará en pantalla lo siguiente:

```
Tokens Totales: 187009
Total de Tokens sin Stopwords: 91055
```

Hasta este punto este ejemplo es igual al anterior, pero ahora, la diferencia es que se cargarán las listas de palabras “*positivas*” y “*negativas*” desde un archivo de texto, cuidando de eliminar los saltos de línea de los archivos de texto, para que se cargue la lista correctamente:

```
# Cargar archivo de palabras positivas
with open('texto_002_palabras_positivas.txt','r') as archivo_en_memoria:
    palabras_positivas = [linea.strip() for linea in archivo_en_memoria]

# Cargar archivo de palabras negativas
with open('texto_003_palabras_negativas.txt','r') as archivo_en_memoria:
    palabras_negativas = [linea.strip() for linea in archivo_en_memoria]
```

Nuestro contador de palabras positivas y negativas lo inicializamos a cero:

```
# Inicializar a cero las cantidades positivas y negativas
numero_de_positivas = numero_de_negativas = 0
```

Ahora buscamos si dentro de nuestro libro hay alguna coincidencia de alguna palabra que está en nuestra lista de palabras positivas y negativas, si es así, incrementamos su respectivo contador:

```
# Buscamos las palabras positivas y negativas dentro del texto
for elemento in lista_final:
    if elemento in palabras_positivas:
        numero_de_positivas = numero_de_positivas + 1
    if elemento in palabras_negativas:
        numero_de_negativas = numero_de_negativas + 1
```

Ahora ya solo falta imprimir el total de palabras positivas y negativas y la polaridad del texto:

```
# Impresión de resultados
print("\nNúmero de palabras positivas encontradas: ", numero_de_positivas)
print("Número de palabras negativas encontradas: ", numero_de_negativas)
print("\nPolaridad del texto:")
polaridad = numero_de_positivas - numero_de_negativas
if polaridad>0: print("Positiva\n")
elif polaridad<0: print("Negativa\n")
else: print("Neutral\n")
```

En pantalla se observará el resultado final:

```
Número de palabras positivas encontradas: 2246
Número de palabras negativas encontradas: 1537

Polaridad del texto:
Positiva
```

Se puede observar que ahora se encontraron muchas palabras tanto “positivas” (2246) como “negativas” (1537) pero aún así, la polaridad se mantuvo “positiva”.



Actividad Práctica: Analiza un texto para determinar su polaridad (positivo, negativo ó neutral) usando TextBlob..

Para este ejemplo ahora se utilizará “TextBlob”, lo cual nos reducirá la cantidad de código y a nivel de programación será mas fácil aunque tendremos que ajustarnos a las restricciones que pueda tener esta biblioteca.



Material Adicional: Para consultar las características de “TextBlob” se puede consultar su página oficial para mayor información: <https://textblob.readthedocs.io/en/dev/>



Advertencia: Aunque “TextBlob” nos facilita la programación, varios autores han indicado que esta biblioteca es muy lenta a comparación de otras técnicas empleadas.

Ahora para este ejemplo se utilizará el texto “Sentimientos de la Nación” de José María Morelos y Pavón (que es de dominio público) y el cual se puede descargar de cualquiera de los siguientes sitios:

- http://www.xumarhu.net/discurso_sentimientos_14_septiembre_1813.txt

- http://sagitario.itmorelia.edu.mx/~rogelio/discurso_sentimientos_14_septiembre_1813.txt

El texto anterior debe de colocarse en el mismo directorio donde haremos nuestro código en Python. Antes de pasar a la programación es necesario ejecutar en una terminal lo siguiente para cargar todos los datos de TextBlob:

- > python3 -m textblob.download_corpora

El primer paso es cargar nuestras bibliotecas necesarias:

```
# Bibliotecas a utilizar
from textblob import TextBlob # Biblioteca para NLP
from textblob.sentiments import NaiveBayesAnalyzer # Clasificador Bayesiano Ingenuo
```

El siguiente paso es cargar a memoria el archivo de texto “*Sentimientos de la Nación*” que descargamos previamente y lo cargamos como un objeto de tipo “*TextBlob*” al mismo tiempo que indicamos que se utilizará el “*Analizador Bayesiano Ingenuo*” (“*Naive BayesAnalyzer*”) para hacer el cálculo de polaridad, el cual ya está entrenado con cierto nivel de probabilidad para detectar la polaridad de un texto:

```
# Cargar archivo de texto a memoria
with open('discurso_sentimientos_14_septiembre_1813.txt', 'r') as archivo_en_memoria:
    texto = archivo_en_memoria.read()

# cargar como objeto de tipo TextBlob y aplicar el Analizador Bayesiano Ingenuo
objeto_textblob = TextBlob(texto, analyzer=NaiveBayesAnalyzer())
```

Ahora se imprimirán en pantalla todos los “tokens” y el número de “tokens” totales. Este paso no es necesario para calcular la polaridad, solo es para monitorear nuestro texto y se puede omitir. Al igual que se puede hacer “*análisis de sentimientos*” con “*TextBlob*”, también se pueden hacer otras operaciones como traducción automática, análisis morfológico, identificación de identidades, etc.:

```
# Detección e impresión de Tokens (no necesario para calcular polaridad)
tokens_de_mi_texto = objeto_textblob.words
print("\nTokens de mi texto:\n", tokens_de_mi_texto)
print("\nTotal de Tokens: ", len(tokens_de_mi_texto))
```

Ya solo falta calcular e imprimir en pantalla la polaridad de nuestro texto, el cual se hace de la siguiente manera:

```
# Calcular e imprimir la polaridad
polaridad = objeto_textblob.sentiment
print("\nPolaridad de mi texto: ", polaridad, "\n")
```

En pantalla se observa lo siguiente:

```
Polaridad de mi texto: Sentiment(classification='pos', p_pos=0.9999995564220755
, p_neg=4.435779255840451e-07)
```

Como se observa, según “*TextBlob*” la polaridad de nuestro texto tiene una alta probabilidad de ser “*Positivo*”, y muy baja de ser “*Negativo*”.

8.2.8 Detección de Similaridades

Otra área importante dentro del NLP es tratar de identificar si 2 textos (que puede ser un conjunto de palabras, frases, textos o incluso libros completos) tiene alguna similitud, y en caso de ser así, cuál sería el grado de similitud entre ambos textos (con el fin de seleccionar de entre varios posibles textos).

La importancia radica en que actualmente se emplean muchas aplicaciones que requieren encontrar similitudes entre frases, algunas de ellas son las siguientes:

- Detección de plagios (identificar la copia de información).
- Sistemas de recomendaciones (detección de gustos similares de películas, música, libros, etc).
- Interpretación de preguntas (detección de palabras y frases para construir chatbots).

Algunas de las técnicas mas usadas son las siguientes:

- Similitud de Coseno.
- Vectorización de textos y comparación.
- Machine Learning y Deep Learning.

A continuación se verán algunos ejemplos prácticos usando estas técnicas.



Actividad Práctica: Comparar 2 vectores para ver su parecido usando la similitud coseno.

Para este primer ejemplo, se realizará una comparación entre 2 vectores (para el siguiente ejemplo se utilizará un texto) esto con el objetivo de ejemplificar esta técnica. Lo primero será cargar las bibliotecas que se utilizarán:

```
# Bibliotecas a utilizar
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
```

Definimos 2 vectores a comparar:

```
# Vectores a comparar
vector_1 = np.array([[1, 1, 0, 1, 1]])
vector_2 = np.array([[0, 1, 0, 1, 1]])
```

Calculamos que tan parecidos son ambos vectores utilizando la similitud del Coseno:

```
# Calcular el parecido por Similitud Coseno
similitud_coseno = cosine_similarity(vector_1, vector_2)
print("\nLa Similitud Coseno entre los vectores es:")
print(similitud_coseno, "\n")
```

En pantalla se observa que tan parecidos son ambos vectores:

```
La Similitud Coseno entre los vectores es:
[[0.8660254]]
```

Si modificamos los vectores y comparamos 2 que son iguales se observará en pantalla lo siguiente:

```
La Similaridad Coseno entre los vectores es:  
[[1.]]
```

Por lo que un valor = “1” es para 2 vectores iguales y un valor = “0” es para 2 vectores que no se parecen en nada. Entre más se acerque el valor del resultado a “1” mayor será la similitud y viceversa.

Una computadora es excelente para buscar y comparar números, pero cuando utilizamos texto se tiene que utilizar alguna técnica para que se puedan resolver los procesos de NLP. La mas sencilla y fácil de usar en una computadora es hacer una lista de palabras del texto, después hacer una relación con un vector numérico, donde cada palabra está representada por un número. De esta manera ya utilizando vectores es más fácil de analizar y compararlos entre ellos. El siguiente ejemplo se utilizará esta técnica.



Actividad Práctica: Comparar 2 textos para ver su parecido usando la similitud coseno.

Nuevamente empezaremos por cargar nuestras bibliotecas necesarias:

```
# Bibliotecas a utilizar  
import numpy as np  
from sklearn.feature_extraction.text import CountVectorizer  
from sklearn.feature_extraction.text import TfidfTransformer  
from sklearn.feature_extraction.text import TfidfVectorizer  
from sklearn.metrics.pairwise import cosine_similarity
```

El siguiente paso es seleccionar las frases a comparar, para lo cual seleccioné 5 que coloqué dentro de una lista:

```
# Frases a comparar  
frases = ["El pony es guinda",  
"El pony es azul",  
"El pony guinda",  
"El pony de colores",  
"Hoy juega Morelia"]
```

Ahora vamos a pasar a vectores (“vectorizar”) y normalizar nuestras frases:

```
# Vectorizar y normalizar  
vectorizer = CountVectorizer()  
frases_vectorizadas = vectorizer.fit_transform(frases)  
print("\nFrases vectorizadas:\n", frases_vectorizadas)
```

Con el código anterior se observará en pantalla una lista de vectores que se crearon a partir de la vectorización:

```

Frases vectorizadas:
(0, 3) 1
(0, 9) 1
(0, 4) 1
(0, 5) 1
(1, 3) 1
(1, 9) 1
(1, 4) 1
(1, 0) 1
(2, 3) 1
(2, 9) 1
(2, 5) 1
(3, 3) 1
(3, 9) 1
(3, 2) 1
(3, 1) 1
(4, 6) 1
(4, 7) 1
(4, 8) 1

```

Aunque no es necesario para el proceso final, se pueden extraer las palabras que se vectorizaron de la siguiente manera:

```

# Extraer los nombres que se vectorizaron
palabras_vectorizar = vectorizer.get_feature_names_out()
print("\nPalabras a Vectorizar: ", palabras_vectorizar)

```

En pantalla se observará una nueva lista, ordenada por orden alfabético que contienen todas las palabras de nuestras frases vectorizadas:

```

Palabras a Vectorizar: ['azul' 'colores' 'de' 'el' 'es' 'guinda' 'hoy'
'juega' 'morelia' 'pony']

```

El siguiente paso tampoco es necesario para el procesamiento final, pero nos ayuda a visualizar cada una de nuestras palabras, así como el número que se le asignó dentro del vector:

```

# Relación entre palabras y su número dentro del vector
relacion = vectorizer.vocabulary_
print("\nRelación palabra valor: ", relacion)

```

En pantalla se observa la relación “*palabra:vector*” conforme van apareciendo, por ejemplo la palabra “*morelia*” le tocó el número 8, por lo tanto, cuando se compare el vector y se encuentren dos números “8” se entenderá que se encontró la palabra “*morelia*” en 2 frases diferentes y por lo tanto hay un grado de similitud entre ambas frases:

```

Relación palabra valor: {'el': 3, 'pony': 9, 'es': 4, 'guinda': 5, 'azul':
0, 'de': 2, 'colores': 1, 'hoy': 6, 'juega': 7, 'morelia': 8}

```

Se procede al cálculo de la matriz de frecuencias de cada elemento del vector de la siguiente manera:

```

# Calcular e imprimir la matriz de frecuencias
matriz = TfidfVectorizer().fit_transform(frases)
print("\nMatriz de Frecuencia:\n", matriz)

```

En pantalla se observa la matriz con sus respectivos valores:

```
Matriz de Frecuencia:
(0, 5) 0.5797475895828805
(0, 4) 0.5797475895828805
(0, 9) 0.40483667370044435
(0, 3) 0.40483667370044435
(1, 0) 0.6614375955758462
(1, 4) 0.5336436873608266
(1, 9) 0.3726424037188896
(1, 3) 0.3726424037188896
(2, 5) 0.711524722134745
(2, 9) 0.4968564026351264
(2, 3) 0.4968564026351264
(3, 1) 0.6160645062907986
(3, 2) 0.6160645062907986
(3, 9) 0.3470799966703276
(3, 3) 0.3470799966703276
(4, 8) 0.5773502691896258
(4, 7) 0.5773502691896258
(4, 6) 0.5773502691896258
```

Ya solo falta comparar una de las frases con todas las demás para observar su grado de similitud. En este ejemplo se va a comparar la primer frase con todas las demás y se imprime el resultado:

```
# Comparar la frase 1 con todas las demás
comparar_1_con_todas = cosine_similarity(matriz[0:1], matriz).flatten()

# Impresión de valores de comparación
print("\nLa Frase 1 comparada contra todas las demás:")
print(comparar_1_con_todas, "\n")
```

En pantalla se observa el resultado obtenido:

```
La Frase 1 comparada contra todas las demás:
[1.          0.61109726 0.81479613 0.28102142 0.          ]
```

El primer "1" que se observa es resultado de la comparación de la frase "1" consigo misma, y por ser iguales el resultado es "1". Después se observan los valores de similitud, donde la mas alta es la "3" (0.814) esto indica que la frase "1" y "3" tienen mayor similitud (lo cual es correcto), que son las siguientes:

- El pony es guinda"
- "El pony es guinda"

El valor mas bajo (0) se observa en la frase "5", lo que indica que no existe similitud entre la frase "1" y la "5" (que también es correcto), que son las siguientes:

- El pony es guinda"
- "Hoy juega Morelia"

8.3 Construcción de ChatBots para Lenguaje Natural

8.3.1 Conceptos sobre ChatBots

Según Sowmya Vajala en su libro "*Practical Natural Language Processing*", un "Chatbot es un sistema interactivo que permite a los usuarios comunicarse utilizando el lenguaje natural" (Vajjala et al., 2020). Las formas de interactuar entre Humanos y Máquinas son normalmente las siguientes:

- A través de interfaces usando texto.
- A través de interfaces de reconocimiento de voz.

A partir del año 2016 se inició la revolución de los “*chatbots*”, ya que las grandes compañías como Apple, Google, Amazon, Microsoft y Facebook fueron introduciendo sus tecnologías de “*chatbots*” en sus diferentes productos, plataformas y ecosistemas tecnológicos. Estas primeras aplicaciones dieron origen a la creación de herramientas, bibliotecas y frameworks para desarrolladores de aplicaciones que utilicen “*chatbots*”, incorporando a la comunidad tecnológica un gran conjunto de aplicaciones de este tipo. Desde la época en que apareció “*Eliza*” y el “*mouse*” en la interfaz gráfica, no se habían visto grandes cambios de interacción entre “*Hombre-Máquina*” hasta nuestros días como sucedió con las interfaces de voz.

8.3.2 Aplicaciones de ChatBots

Algunas de las aplicaciones mas utilizadas actualmente de los "ChatBots" son las siguientes:

- Búsqueda de información.
- Ventas y Compras.
- Atención a clientes.
- Asistentes personales.
- Asistentes especializados (médicos, legales, etc.).
- Activación y control de dispositivos(Domótica).

8.3.3 Ubicación de ChatBots

Los “*chatbots*” pueden ir desde los más sencillos hasta los más complejos, de acuerdo a la cantidad de información que necesitan procesar y se pueden localizarse básicamente en:

- Dispositivos físicos pequeños (como celulares) y generalmente son muy básicos.
- Utilizan dispositivos físicos para capturar la información, pero es enviada a la nube para su procesamiento y este tipo de “*chatbots*” resultan mas completos y complejos.

8.3.4 Clasificación de ChatBots

Los “*chatbots*” de acuerdo a su funcionamiento los podemos clasificar en:

- Preguntas Respuestas Frecuentes, mas conocidos como **FAQ** (“*Frequently Asked Questions*” por su significado en inglés): Están diseñados solamente para identificar muy pocas preguntas, muy específicas y poderlas contestar también de una sola manera muy específica. Se utilizan para resolver problemas muy específicos o dar información sobre algo en particular, por ejemplo cuando se pregunta por el precio de un artículo durante una compra.
- **Flujo**: Son mas complejos que los FAQ y en éstos, las conversaciones son mas fluidas y dependen de las preguntas que hagan los usuarios. Se utilizan mucho para ordenar pedidos donde se utilizan varias secuencias de preguntas con el objetivo de completar un pedido.
- **Sin Fin**: Se utilizan mas en el área de entretenimiento y se utilizan para entablar conversaciones sobre varios tópicos, saltado de unos a otros, con el objetivo de mantener la conversación abierta y continua. Son muy usados por ejemplos en los asistentes personales como Siri, Alexa, Google Assistant ó Cortana.

A continuación algunos ejemplos prácticos de “*chatbots*”.



Actividad Práctica: Hacer un “*chatbot*” básico en un ciclo infinito para esperar por preguntas y contestar.

Para hacer este ejemplo utilizaremos la biblioteca “*chatterbot*”, la cual nos permite de manera fácil construir “*chatbots*” muy simples que pueden ir mejorando y entrenando. Como primer paso cargaremos la única biblioteca que necesitamos para este ejemplo:

```
# Bibliotecas a utilizar
from chatterbot import ChatBot
```



Advertencia: La biblioteca “*chatterbot*” no es compatible con la versión mas nueva de Python (3.10.x) por lo cual se recomienda instalar la versión anterior desde consola con la instrucción siguiente: `pip3 install chatterbot==1.0.4`

El siguiente paso es declara nuestro chatbot:

```
# Declarar mi Bot
mi_chat = ChatBot("Pruebas")
```

Y por último, crear un ciclo infinito donde la computadora espera por una pregunta ó comentario del usuario para después imprimir una respuesta del “*chatbot*”, regresando nuevamente a esperar por otra pregunta, y esto se hace de la siguiente manera:

```
# Bot en ciclo para esperar pregunta y contestar
print("\n")
while True:
    pregunta = input("Yo: ")
    respuesta = mi_chat.get_response(pregunta)
    print("Bot: ", respuesta)
```

En pantalla se puede ir observando las interacciones entre el usuario y el “*chatbot*”:

```
Yo: Hi
Bot: Hi
Yo: How are you?
Bot: Hi
Yo: Are you ok?
Bot: How are you?
Yo: Fine and you?
Bot: How are you?
Yo: █
```

Este primer “*chatbot*” no tiene entrenamiento ni tampoco respuestas concretas asignadas, por lo cual sus contestaciones son muy malas y vagas pero es un primer intento muy fácil de realizar de un “*chatbot*” y con muy poco código. A la biblioteca “*chatterbot*” se le han agregado varias funcionalidades con el fin de poder entrenar al “*chatbot*” al mismo tiempo que puede aprender, pero varias de estas funcionalidades no funcionan adecuadamente con Python 3.10.x por ciertas incompatibilidades que esperemos puedan resolver los desarrolladores próximamente. A continuación otro ejemplo práctico de “*chatbot*” de tipo FAQ (orientado a Pregunta-Respuesta).



Actividad Práctica: Hacer un “*chatbot*” que respondan preguntas específicas sobre una pizzería.

Para este ejemplo sólo utilizaremos el lenguaje Python básico, es decir, no cargaremos ninguna biblioteca en específico. Este “*chatbot*” como es de tipo “*Pregunta-Respuesta Frecuente*” (FAQ), entonces tenemos que definir las preguntas que podrá resolver, así como las respuestas que daría a tales preguntas, las cuales colocaremos en 2 listas y que son:

```
# Definir las preguntas que responderá el chatbot
preguntas = [
    "de cuáles pizzas tienen",
    "cuánto cuestan",
    "tienen servicio a domicilio",
    "cuál es su teléfono",
    "adios",
]
```

```
# Definir las respuestas del chatbot
respuestas = [
    "Peperoni, Hawaiana y Mexicana",
    "$150",
    "Si",
    "443-312-15-70",
    "Fué un gusto atenderte!!!\n",
]
```

El siguiente paso es imprimir un mensaje de arranque del “*chatbot*” y se utilizará una variable booleana para determinar el inicio y fin del chat:

```
# Mensaje de inicio del chatbot
print("\nHola! Soy tu bot de servicio, a tus órdenes:\n")
conversacion = True
```

Por último se genera un ciclo que funcionará mientras la conversación esté activa (`conversacion == True`). Dentro del ciclo detenemos el programa para preguntarle al usuario (`input`) cuál es su pregunta. La respuesta del usuario la convertimos a minúsculas para compararla contra nuestra lista de preguntas que también está en minúsculas y así pueda encontrar alguna posible pregunta.

Si la respuesta del usuario es “*adios*” cambiamos el valor de nuestra variable booleana para poder terminar la conversación (`conversacion_usuario == False`). El siguiente paso es buscar dentro de nuestra lista si hay alguna coincidencia, y si es así, entonces procedemos a imprimir la respuesta que tenemos asignada a esa pregunta y el programa queda de la siguiente manera:

```
# Ciclo para reguntar y contestar
while (conversacion == True):
    respuesta_usuario = input("Usuario: ")
    respuesta_usuario = respuesta_usuario.lower()
    if (respuesta_usuario == "adios"):
        conversacion = False
    contador = 0
    for frase in preguntas:
        if(frase == respuesta_usuario):
            print("Bot: ", respuestas[contador])
        else:
            contador = contador + 1
    if (contador == 5):
        print("Bot: En qué puedo ayudarte?")
```

En caso de que la pregunta del usuario no se encuentre dentro de nuestra lista, entonces se procederá a imprimir un mensaje en pantalla “*En qué puedo ayudarte?*”. En pantalla se observará el desarrollo de la conversación hasta cuando finaliza, cuando el usuario escribe “*adios*”:


```
Hola! Soy tu bot de servicio, a tus órdenes:

Usuario: Hola que tal!
Bot: En qué puedo ayudarte?
Usuario: Me llamo Rogelio
Bot: En qué puedo ayudarte?
Usuario: de cuáles pizzas tienen
Bot: Peperoni, Hawaiana y Mexicana
Usuario: cuánto cuestan
Bot: $150
Usuario: tienen servicio a domicilio
Bot: Si
Usuario: cuál es su teléfono
Bot: 443-312-15-70
Usuario: adios
Bot: Fué un gusto atenderte!!!
```

Hay que hacer notar que en este ejemplo, si el usuario escribe su pregunta un poco diferente de la pregunta que tenemos nosotros, no podrá encontrarla (deben escribirse exactamente igual), por lo que es necesario implementar un análisis de similitud de texto (como el que se hizo en ejercicios anteriores) para poder encontrar preguntas muy similares a la que el usuario quiere hacer y de esta manera poder contestar adecuadamente.

Capítulo

Herramientas y Aplicaciones

9

Objetivo: Conocer algunas de las herramientas que existe para ciencia de datos.

“Una computadora puede ser llamada “inteligente” si logra engañar a una persona haciéndole creer que es un humano”

Alan Turing

El inglés Alan Mathison Turing (1912-1954) fue un matemático que formalizó el concepto de algoritmo a través de la "Máquina de Turing". Fue el encargado de lograr decodificar los mensajes de la máquina alemana "Enigma" durante la Segunda Guerra Mundial. Se le considera el "Padre de la Inteligencia Artificial" por proponer la "Prueba de Turing" para considerar si una máquina es inteligente.



En este último capítulo se verán algunas de las herramientas mas avanzadas de Machine Learning que han desarrollado algunas de las grandes empresas tecnológicas (IBM, Google, Microsoft y Amazon).

Estas herramientas son de tipo “nube”, con un sistema de acceso bajo demanda, bajo el modelo “Software as a Service” principalmente, aunque también incluyen la parte de almacenamiento, hosting y renta completa de infraestructura de red, con herramientas ya pre-construidas por las mismas compañías, con la opción de poder desarrollar y montar en la nube tus propias herramientas.

Por último se verán algunas aplicaciones impresionante que se están desarrollando en este preciso momento y que hace pocos años nos parecían de “Ciencia Ficción”, pero que gracias a la gran cantidad de datos que se están logrando acumular, el avance en el desarrollo de nuevos algoritmos y el aumento en el poder de cómputo, logran avances sorprendentes.

Muchas de estas herramientas siguen en desarrollo y experimentación a tal grado y con tanto desempeño que algunas no las han dejado sacar al dominio público ó lo han hecho de manera restringida debido a los posibles impactos en la sociedad, la cultura y la tecnología a las cuales puedan afectar de maneras impredecibles.

9.1 Herramientas en la Nube

9.1.1 IBM Watson Studio

"Watson" es un conjunto de herramientas y plataformas en la nube desarrolladas por IBM que pueden ser consumidas bajo un esquema de tipo "Software as a Service" con un costo para los usuarios utilizando principalmente lenguaje natural como interface.

"Watson" es descendiente del proyecto "Deep Blue" que derrotó a Gary Kasparov en 1997 y que posteriormente se integró al proyecto "Watson" que derrotó a los campeones mundiales de Jeopardy, Brad Rutter y Ken Jennings en 2011 (de ahí viene el nombre de este conjunto de herramientas).

En el año 2013 IBM lo lanza como una aplicación comercial donde inicialmente se aplicó al área de medicina, pero desde entonces se le han integrado varios módulos para que el sistema pueda resolver preguntas realizadas en lenguaje natural por los usuario en diferentes áreas.

En el 2016 IBM anunció una inversión de \$300 millones de dólares en investigación sobre Ciencia de Datos y en 2018 se hizo una actualización de su plataforma "Watson" e introdujo un conjunto de aplicaciones y servicios de Machine Learning orientado principalmente a "Business Intelligence". A partir de este momento la renombró como "IBM Watson Studio" (figura 78).

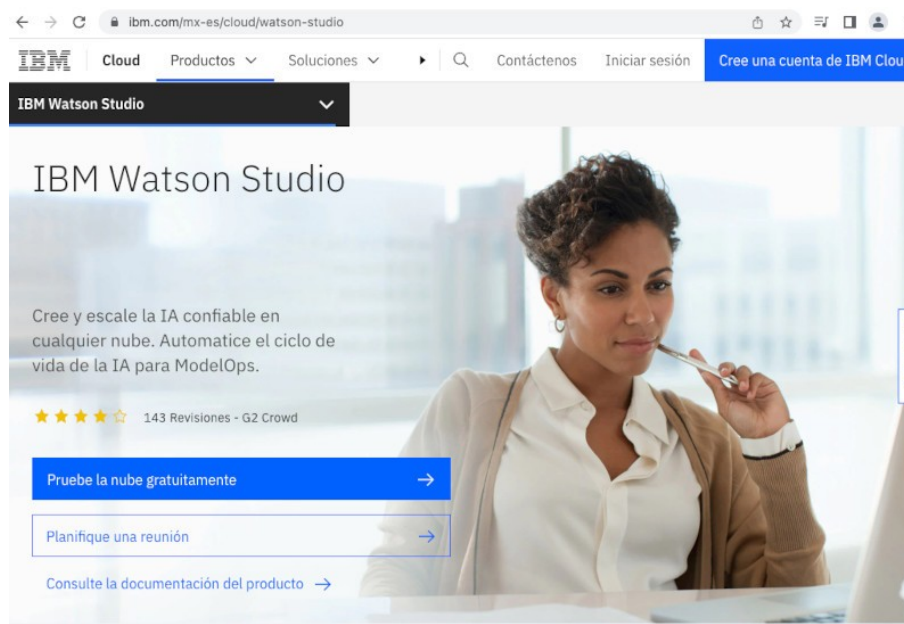


Figura 78: Página oficial de "IBM Watson Studio"

Algunas de las aplicaciones que incluye "IBM Watson Studio" son las siguientes:

- Herramientas para crear tus propios modelos de Machine Learning.
- Modelos de Machine Learning ya entrenados para ciertas aplicaciones.
- Módulos de Data Mining para análisis de textos y minería de datos (SPSS Modeler).
- Plataforma para una visualización inteligente de datos (Cognos Analytics).
- Manejo de Clústeres en la nube con modelos de Machine Learning (Analytics Engine).



Material Adicional: Para mayor información de “IBM Watson Studio” se puede consultar su página oficial: <https://www.ibm.com/mx-es/cloud/watson-studio>

Se puede uno registrar de manera gratuita y probar de manera muy restringida algunas herramientas, ya que esta herramienta tiene un costo por su uso (figura 79), y según su página oficial sus costos son los siguientes:

	Standard	Más popular Premium
	USD 10*/usuario/mes	USD 40*/usuario/mes

Figura 79: Precios de uso de "IBM Watson Studio".

Donde el paquete “Premium” debido al costo incluye mas herramientas.



Material Adicional: Se puede usar “IBM Watson Studio” de manera restringida si se registra en su página oficial: <https://dataplatfrom.cloud.ibm.com/registration/stepone>

9.1.2 Google Cloud AutoML

Google es uno de los pioneros de las herramientas en línea y líder en la industria de aplicaciones “online”, por lo tanto, no podría quedarse atrás en la carrera tecnológica por ofrecer una plataforma orientada a Ciencia de Datos con Inteligencia Artificial.

En el año 2017, durante la Conferencia Google I/O anunció la creación de la división “Google AI”, la cual se dedicaría a investigar y aplicar nuevas tecnologías y herramientas que se incluirían próximamente dentro de su ya de por si gran suite de aplicaciones en línea. Con el tiempo se fueron agregando mas componentes y actualmente algunos de ellos son los siguientes:

- Vertex AI: Conjunto de modelos de Machine Learning para aplicaciones específicas.
- AutoML Tabular: Para crear y desplegar modelos de Machine Learning.
- AutoML Image: Se aplica a imágenes para la detección de objetos y poder extraer información.
- AutoML Video: Se aplica a videos para la detección de objetos y poder extraer información.
- AutoML Text: Análisis de textos usando Machine Learning.
- AutoML Translation: Detecta idiomas y traduce entre ellos.

Al mismo tiempo que fue incorporando herramientas, también ha desarrollado investigación para aumentar el poder de cómputo necesario para soportar este tipo de aplicaciones de manera masiva. Google ha incrementado su poder de cómputo y ha conectando entre sí sus propios Clústers y Datacenters.

Por ejemplo, la herramienta de Google “Vertex AI”, logra detectar dentro de una imagen no solamente una casa, si no el tipo de diseño que tiene la casa (figura 80). Esto puede ser muy importante para un diseñador que necesite clasificar e identificar casas (Google_01, 2022).



Figura 80: Google Vertex AI para identificar estilos de casas.

Otra herramienta de Google “Vertex AI” puede identificar y clasificar mensajes de los clientes para poderlos asignar a las diferentes áreas y personas de una empresa (figura 81), para que los puedan atender de manera más fácil, ahorrando tiempo y dinero:



Figura 81: Google Vertex AI para clasificación de mensajes.

Estos son algunos de los precios que maneja Google para su aplicación Google “Vertex AI” (figura 81), son precios para procesamiento de imágenes, hay otros precios diferentes para procesamiento de videos, texto y datos tabulares (Google_02, 2022):

Operación	Precio por hora de nodo (clasificación)	Precio por hora de nodo (detección de objetos)
Formación	3,465 USD	3,465 USD
Entrenamiento (modelo perimetral en el dispositivo)	18,00 USD	18,00 USD
Despliegue y predicción online	1,375 USD	2,002 USD
Predicción por lotes	2,222 USD	2,222 USD

Figura 82: Precios de Google Vertex AI para procesamiento de imágenes.

9.1.3 Azure Machine Learning

“Azure” es una plataforma en la nube desarrollada por Microsoft y presentada en febrero del 2010 inicialmente con el nombre de "Windows Azure" y posteriormente renombrada como "Microsoft Azure" en el 2014. Inicialmente Microsoft presentó Servicios de tipo Infraestructura ó "IaaS" ("Infrastructure as a Service" por sus siglas en inglés) con máquinas virtuales de Microsoft Windows. Posteriormente fueron agregando mas componentes de la Suite de Microsoft para trabajar en la nube.

En el año 2014 se presentó el "Azure Machine Learning" (mas conocido como "Azure ML"), el cual es un conjunto de herramientas y frameworks para que el usuario pudiera construir sus propios servicios de Machine Learning. Posteriormente Microsoft presentó "Azure Cognitive Services", el cual es un conjunto de API's y SDKs pre-construidos para que los desarrolladores pudieran implementar reconocimiento de voz, sintetización de voz, visión por computadora, reconocimiento de rostros, procesamiento del lenguaje natural, traducción de idiomas y servicios orientados inteligencia de negocios.

Para integrar de manera gráfica Microsoft lanzó "Azure Machine Learning Studio" (figura 83), el cual es una suite con una interfaz sencilla en donde se pueden arrastrar e interconectar componentes de Machine Learning (“drag and drop”) así como los recursos necesarios para construir modelos predictivos (Microsoft_01, 2022).

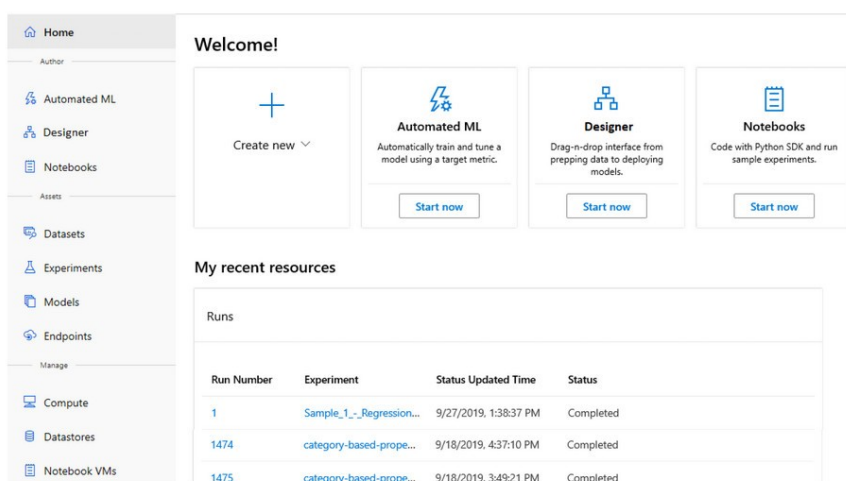


Figura 83: Pantalla de Azure Machine Learning Studio.

Estos son algunos precios (figura 84) que maneja Microsoft para “Azure Machine Learning” (Microsoft_02, 2022):

Instance	vCPU(s)	RAM	Linux VM Price	Machine Learning Service Surcharge	Pay As You Go Total Price	1 year reserved total price	3 year reserved total price
D2 v3	2	8 GiB	\$0.096/hour	\$0/hour	\$0.096/hour	\$0.058/hour ~40% savings	\$0.037/hour ~62% savings
D4 v3	4	16 GiB	\$0.192/hour	\$0/hour	\$0.192/hour	\$0.115/hour ~40% savings	\$0.074/hour ~62% savings
D8 v3	8	32 GiB	\$0.384/hour	\$0/hour	\$0.384/hour	\$0.229/hour ~40% savings	\$0.148/hour ~62% savings

Figura 84: Precios de "Azure Machine Learning".

9.1.4 Amazon Machine Learning

Amazon, el líder en servicios de nube empezó con pequeños servicios en el año 2000, los cuales consistían en API's para conectarse a su red de servidores. En el año 2004 se presenta oficialmente ya con el nombre de "*Amazon Web Services*", la cual ya incluía un conjunto de servicios de tipo "*IaaS*".

En el 2006 lanza "*Simple Storage Service*" ("Servicios de Almacenamiento en Nube") y posteriormente "*Amazon Elastic Compute Cloud*" el cual es un servicio "*On-Demand*", en donde los usuarios pueden solicitar mas recursos (almacenamiento y procesamiento) de manera dinámica mientras están trabajando, lo cual creó un precedente que se mantiene hasta nuestros días en este tipo de servicios.

En el año 2015 Amazon anuncia su sistema de servicios de Machine Learning y desde entonces empieza a integrar nuevos servicios de este tipo. Algunas de las aplicaciones que integra Amazon en sus herramientas de Machine Learning e Inteligencia Artificial son las siguientes (Amazon_01, 2022):

- Amazon Rekognition: Análisis de imágenes y videos.
- Amazon Lookout for Visión: Identificación visual de fallas en componentes de varios tipos en líneas de producción.
- Amazon Textract: Análisis masiva de textos.
- Amazon Comprehend: Análisis del contenido de textos.
- Amazon Lex: Crear chatbots y agentes virtuales.
- Amazon Transcribe: Reconocimiento de voz y su transcripción.
- Amazon Translate: Traducción automática entre idiomas.
- Amazon Forecast: Crear modelos de predicción.

Las herramientas de Amazon para Machine Learning crecen muy rápido, por ejemplo (figura 85), su herramienta para detección de rostros ya puede detectar el tipo de persona (hombre ó mujer), si los ojos están abiertos, el estado anímico (feliz, triste, etc), y expresión vocal (sonriendo, llorando, etc.), además de otras tantas características (Amazon_02, 2022):

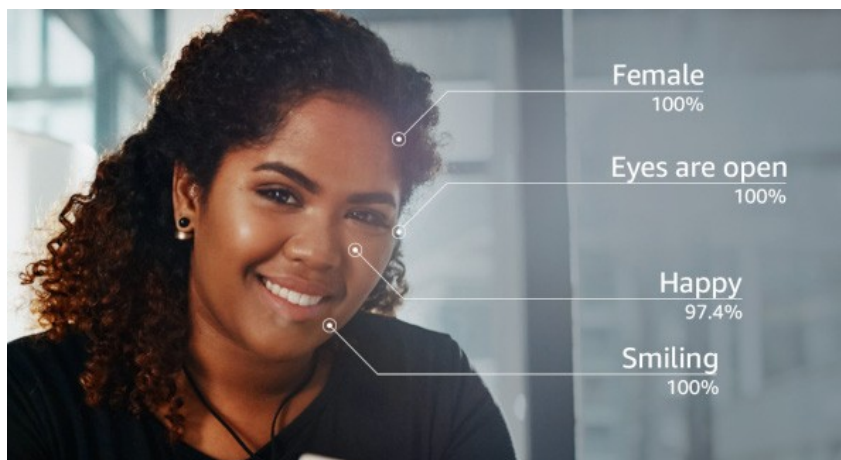


Figura 85: Ejemplo de "*Amazon Rekognition*".

El precio de "*Amazon Machine Learning*" es de acuerdo al aplicación solicitada, así como el espacio de almacenamiento, poder de cómputo y ancho de banda utilizado. A continuación un ejemplo del costo de "*SageMaker Studio*" (figura 86), el cual es la herramienta para crear e implementar modelos de Machine Learning dentro de AWS:

Instancias estándar	CPU virtual	Memoria	Precio por hora
ml.t3.medium	2	4 GIB	0,05 USD
ml.t3.large	2	8 GIB	0,10 USD
ml.t3.xlarge	4	16 GIB	0,20 USD
ml.t3.2xlarge	8	32 GIB	0,399 USD

Figura 86: Precios de "SageMaker Studio" de Amazon.



Material Adicional: Para calcular el costo preciso de algunos servicios, Amazon dispone de una calculadora en línea: <https://calculator.aws/#/addService/SageMaker>

9.2 Aplicaciones Avanzadas

9.2.1 GPT

GPT-3 (*Generative Pre-trained Transformer 3* por sus siglas en inglés) es una herramienta que sirve para construcción de textos simulando la redacción humana utilizando una técnica específica de Deep Learning llamada *Transformers*. Fue desarrollado por investigadores de la *OpenAI* y presentada oficialmente el 28 de mayo del 2020 por medio del artículo *Language Models are Few-Shot Learners* (Brown et al, 2020).

Está formada por 175 mil millones de parámetros y llega a producir textos que ya son difíciles de determinar si fueron escritos por un humano ó una máquina.

En junio del 2022, Almira Osmanovic Thunström que trabaja en el *"Departamento de ePsychiatría"* en *"Sahlgrenska University Hospital"* de Gotemburgo Suecia, utilizó GPT-3 para generar un artículo que hablaba sobre el mismo GPT-3 y lo envió para su publicación. Actualmente está en revisión y de ser publicado, sería el primer artículo científico, aceptado y revisado para su publicación que haya sido escrito por una inteligencia artificial con una mínima ayuda humana (Osmanovic, 2022).

Se puede acceder a GPT-3 por medio de un API (con previo registro para obtener un *token*) a través de varios lenguajes, ó también se pueden hacer pruebas directamente sobre su plataforma en línea donde se pueden seleccionar la actividad a realizar y posteriormente introducir un texto para que la herramienta genere el contenido.

Un ejemplo puede ser que vayamos a escribir un ensayo (herramienta *Essay outline* de GPT-3) para una investigación, entonces se tecléa que queremos hacer un ensayo sobre *"Nikola Tesla y su contribución a la tecnología"*, por lo cual escribimos lo siguiente:

- *"Create an outline for an essay about Nikola Tesla and his contributions to technology"*:

Y en pantalla observaremos el resultado que nos arroja GPT-3 (figura 87):

Essay outline

Generation

Generate an outline for a research topic.

Prompt

Create an outline for an essay about Nikola Tesla and his contributions to technology.

Settings

Engine	text-davinci-002
Max tokens	150
Temperature	0
Top p	1.0
Frequency	0.0
penalty	
Presence penalty	0.0

Sample response

1. Introduce Nikola Tesla and his work in the field of electricity and magnetism.
2. Discuss Tesla's contributions to the development of alternating current (AC) electricity.
3. Describe Tesla's development of the Tesla coil and its impact on wireless technology.
4. Discuss Tesla's experiments with X-rays and their implications for the future of medical technology.
5. Conclude with a discussion of Tesla's legacy and his impact on the development of modern technology.

Figura 87: Ejemplo de GPT-3 para generar un ensayo.



Material Adicional: Se puede consultar el resultado de más ejemplos de GPT-3 en su página Web: <https://beta.openai.com/examples>

9.2.2 GitHub Copilot

“*GitHub Copilot*” es una herramienta para asistir a los desarrolladores durante la programación por medio de la autocompletación de código. Es una herramienta presentada el 29 de junio del 2021 y que utiliza “*OpenAI Codex*”, la cual es una versión modificada de GPT-3 y que fue licenciada a Microsoft para ser usada en GitHub. Fue entrenada con varios terabytes de código fuente almacenados en la plataforma de GitHub.

Según estadísticas realizadas por sus desarrolladores, GitHub Copilot es capaz de completar código Python al primer intento en 43% de las ocasiones. Está disponible para ser instalado como un Plug-in para algunos editores de texto como son:

- Visual Studio.
- Visual Studio Code.
- Neovim.
- JetBrains IDEs.

Un ejemplo de demostración que se presenta en su propia página oficial de GitHub Copilot es, dado el siguiente segmento de código escrito por el desarrollador (figura 88):

```

1 package main
2 type Run struct {
3     Time int // in milliseconds
4     Results string
5     Failed bool
6 }
7 // Get average runtime of successful runs in seconds
8 func averageRuntimeInSeconds(runs []Run) float64 {

```

Figura 88: Código introducido por el desarrollador

El “*GitHub Copilot*” genera automáticamente (autocompleta) el siguiente código, de acuerdo al anterior que escribió previamente el desarrollador (figura 89):

```

9     var totalTime int
10    var failedRuns int
11    for _, run := range runs {
12        if run.Failed {
13            failedRuns++
14        } else {
15            totalTime += run.Time
16        }
17    }
18    averageRuntime := float64(totalTime) / float64(len(runs) - failedRuns) / 1000
19    return averageRuntime
20 }

```

Figura 89: Código autocompletado por *GitHub Copilot*



Material Adicional: El ejemplo anterior se extrajo de la página oficial de “*GitHub Copilot*” y ahí mismo se puede consultar mayor información: <https://github.com/features/copilot>



Material Adicional: Se puede acceder de manera gratuita como estudiante a la plataforma “*GitHub Copilot*”, ó también se puede tener una prueba gratuita por 60 días pero después se tiene que pagar un costo de \$10 dólares mes ó \$100 dólares año: <https://education.github.com/>

9.2.3 DALL-E

Es un software desarrollado por “*OpenAI*” que se basa en la arquitectura de “*Transformers*” y que tiene por objetivo crear imágenes a partir de descripciones dadas por el usuario de manera escrita en la terminal. Su nombre es un acrónimo formado a partir de las palabras “*Salvador Dali*” y “*WALL-E*”, el cual usa 12 mil millones de parámetros de “*GPT-3*”. Fue publicada la versión 1 en enero del 2021 y en el 2022 la versión 2 (pero no se liberó código fuente) y fue puesta a disposición una versión Demo al público, que posteriormente fue cerrada por cuestiones éticas y solo dejaron un pequeño demo controlado con algunos ejemplos. En ciertos momentos ha sido abierto una API para que algunos programadores pudieran hacer pruebas con esta aplicación.

“*DALL-E*” Es un sistema que *OpenAI* denominó “*Inteligencia Artificial Multimodal*”, en donde fusiona Visión por Computadora y Procesamiento del Lenguaje Natural.

Puede generar ambientes que parecen surrealistas, ya que puede combinar imágenes reales con ficticias creando composiciones visuales únicas. A continuación se muestran 3 ejemplos de las

imágenes que genera de acuerdo a cada uno de los textos de entrada que se le proporcionaron a “DALL-E”:

- “a cat wearing a striped sweater and a polka-dot hat” (figura 90).
- “a panda making latte art” (figura 91).
- “a horse piggybacking an astronaut” (figura 92).



Figura 90: Gato generado por DALL-E mini



Figura 91: Panda generado por DALL-E mini



Figura 92: Astronauta generado por DALL-E mini

Existe una versión “mini” de “DALL-E” abierta al público y donde se pueden hacer pruebas y comprobar los resultados. Es un sitio Web donde uno puede escribir un texto, hacer click en “Run”, esperar un momento (todo el procesamiento se hace en la nube) y posteriormente aparecerá en pantalla un conjunto de 9 imágenes posibles a partir la cadena de texto introducida. Por ejemplo, introduciremos la siguiente cadena y observamos el resultado (figura 93):

- “pony flying in morelia”.

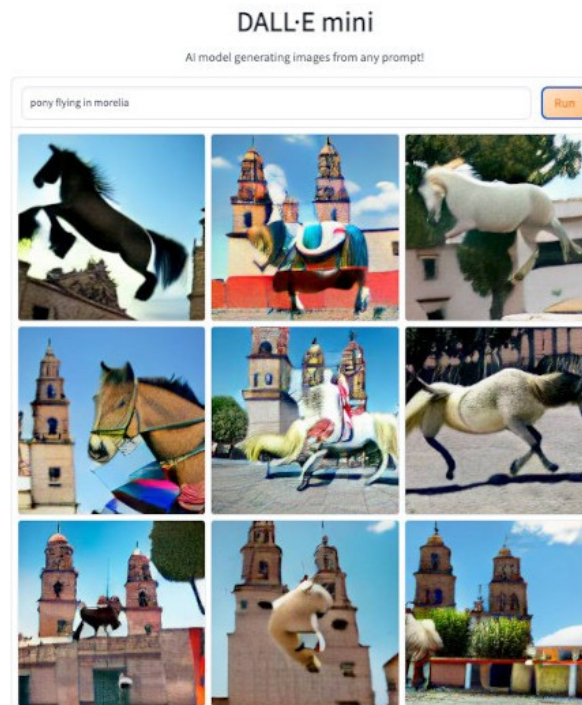


Figura 93: Imágenes generadas por DALL-E mini



Material Adicional: El sitio para probar “*DALL-E mini*” es el siguiente: <https://huggingface.co/spaces/dalle-mini/dalle-mini>



Material Adicional: El proyecto “*DALL-E mini*” con código completo se encuentra en: <https://github.com/borisdayma/dalle-mini>

La versión 2 de “*DALL-E*” ya genera mejores composiciones e imágenes foto-realistas de personas y lugares, pero por el momento no es accesible a todo el público y solamente puede solicitar el acceso a través de una lista de espera. Un ejemplo de lo que genera este software es dado el siguiente texto genera la siguiente imagen (figura 94):

- “*A astronaut riding a horse in a photorealistic style*”.



Figura 94: Composición generada por “*DALL-E 2*”



Material Adicional: La imagen anterior es de prueba y se puede observar en la página oficial del proyecto “*DALL-E 2*”: <https://openai.com/dall-e-2/>



Material Adicional: Para apuntarse a la lista de espera y probar “*DALL-E 2*” se puede hacer mediante registro en su página oficial: <https://labs.openai.com/waitlist>

9.2.4 Imagen

Continuando con las investigaciones derivadas del proyecto “*DALL-E*”, el 23 de mayo del 2022 se presentó el artículo “*Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*”, por un conjunto de investigadores del equipo “*Google Brain*”, que es parte de la compañía Google (Saharia et al., 2022).

El proyecto ha sido denominado “*Imagen*” y es un modelo llamado de “*difusión*”, al que se le introduce un texto por medio de una terminal y que puede generar imágenes de baja calidad que luego por “*difusión*” (con ayuda de miles de imágenes y técnicas de Deep Learning) puede generar finalmente una imagen con un grado de foto-realismo muy alto.

Un ejemplo de lo que genera este software es dado el siguiente texto genera la siguiente imagen (figura 95):

- “A robot couple fine dining with Eiffel Tower in the background”.

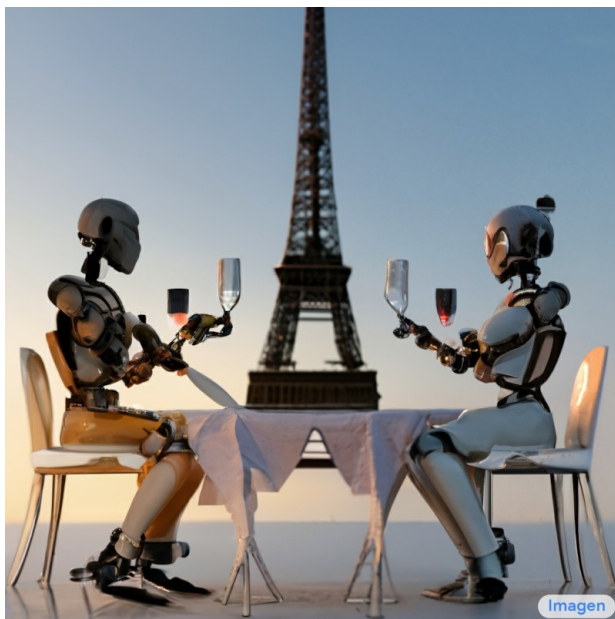


Figura 95: Composición generada por "Imagen"



Material Adicional: La imagen anterior es de prueba y se puede observar su página oficial: https://imagen.research.google/main_gallery_images/a-robot-couple-fine-dining.jpg

9.2.5 LaMDA

El "Modelo de Lenguaje para Aplicaciones de Diálogo", más conocido como "LaMDA" ("Language Model for Dialogue Applications" por sus siglas en inglés), es una aplicación desarrollada por Google para entablar conversaciones hombre-máquina de gran realismo.

Su primera versión fue presentada en el 2021 durante el Google I/O, y en junio del 2022 fue presentada la segunda versión denominada LaMDA 2.

Tiene la característica de extraer textos de numerosas fuentes para poder entablar diálogos de acuerdo a cómo se vaya presentando la conversación, logrando un intercambio fluido de temas de conversación.

Tal como se comentó en el capítulo 7, LaMDA cobró fama internacional cuando el 11 de marzo del 2022 sale publicado en el Washington Post un artículo donde Blake Leomoine (ingeniero de Google) afirma que le hizo varias pruebas (a través de conversaciones) al "chatbot" inteligente "LaMDA 2". Después de realizadas las pruebas (junto con un colega suyo que también trabaja en Google, pero se mantuvo en el anonimato) afirmó que "LaMDA" tiene "vida y sentimientos propios". Lemoine afirma que el programa se comporta como un niño de 7 u 8 años que sabe de física y muchas cosas más (Tiku, 2022).

A continuación un ejemplo del tipo de conversaciones que puede entablar "LaMDA" (figura):

If my washing machine and tv are both malfunctioning at the same time what could the reason be?

It could be the power surge from the washing machine may have fried your tv. If it's a power surge problem it may be time to get a new washing machine. Or new TV.

Figura 96: Un ejemplo de la conversación que puede entablar LaMDA 2.

9.2.6 Stable Diffusion

El 22 de agosto del 2022 (un día antes de cerrar este libro para su revisión) se anunció el lanzamiento de "Stable Diffusion", una aplicación de generación de imágenes a través de textos, creada por "Stability AI".

Por ser una herramienta tan nueva apenas empieza a ser probada por la comunidad y está disponible una versión gratuita para hacer pruebas y hay una versión beta con costo a través de la plataforma DreamStudio.

Las primeras pruebas indican que puede generar una gran calidad de imágenes, e incluso algunos afirman que entiende mejor los textos de entrada y genera mejor y de mayor calidad las imágenes, mejores incluso que "DALL-E 2" e "Imagen". Un ejemplo obtenido de su página Web (Stable_01, 2022) es el siguiente (figura):



Figura 97: Ejemplo 1 de generación de imágenes con "Stable Diffusion".

Nuevamente se probó esta herramienta con el mismo texto que se usó de prueba con "DALL-E Mini" para tratar de compararlos con esta nueva herramienta en su sitio de prueba (Stable_01, 2022) con la siguiente cadena:

- "pony flying in morelia".

Y el resultado fue el siguiente (figura):



Figura 98: Ejemplo 2 de generación de imágenes con "Stable Diffusion".

Como se puede observar, genera imágenes de gran calidad y en poco tiempo de cómputo.

El hombre desde la época de las cavernas siempre quiso plasmar sobre la piedra las ideas, pensamientos y acciones de su época para ser recordadas, y hoy en la actualidad la “*Ciencia de Datos*” lleva eso al extremo, a tratar de registrar cada segundo de nuestra existencia, generándose millones de datos por todo el mundo.

Sin duda, vivimos la “*edad de los datos*”, donde existen muchos dispositivos que están registrando y trasladando miles de datos a miles de servidores por todo el Internet, a los que llamamos “*la nube*”, la cual contiene miles de procesadores y sistemas de almacenamiento, registrando cada cosa posible de nuestro mundo. Es por eso la importancia de conocer de esta nueva área llamada “*Ciencia de Datos*”.

Otra área de gran crecimiento es el “*Internet de las Cosas*”, la cual se encarga de estudiar todos estos pequeños dispositivos que están registrando información por todos lados, desde la temperatura hasta nuestro pulso cardiaco por medio de nuestros relojes de muñeca que nos monitorean todo el día. El “*Internet de las Cosas*” ha impulsado también fuertemente la creación de esta gran cantidad de datos que se están generando, ya que cada vez estos dispositivos son mas pequeños, mas potentes y mas baratos.

Otra área que domina el espectro mundial de la tecnología es la “*Inteligencia Artificial*”, la cual ha permitido crear aplicaciones que hace pocos años sólo se veían en las películas de ciencia ficción y que se creían poco probables de que se llevaran a cabo. Lo increíble es que ya vivimos inmersos dentro de muchas aplicaciones que usan la inteligencia artificial como es el reconocimiento de voz y rostro, recomendaciones de películas, mapas en nuestros celulares que nos guían a nuestro destino, motores de búsqueda que revisan gran parte del Internet para encontrar la información que necesitamos y cientos de aplicaciones que cada día aparecen en nuestras vidas y que se van incorporando a nuestro día a día, a veces sin darnos cuenta.

Otro punto importante para que las tecnologías anteriores pudieran crecer, es el aumento en el poder de cómputo de todos nuestros dispositivos, desde los grandes datacenters, pasando por nuestras computadoras portátiles hasta nuestros relojes y dispositivos personales. Ahora ya empezamos a ver la integración en otros dispositivos de nuestra casa y de nuestros automóviles, y próximamente dispositivos tan pequeños que sean introducidos dentro de nuestro cuerpo para monitorear nuestro estado de salud. La próxima revolución de la “*Computación Cuántica*” que ya está en marcha, promete aumentar de manera exponencial el poder de cómputo para hacer posibles cosas aún mas increíbles, aunque por estar apenas en proceso de investigación, todavía no sabemos sus implicaciones, alcances y efectos reales.

La actual “*Revolución de los Datos*” no existiría si no se hubieran mejorado los algoritmos de “*Inteligencia Artificial*”, ni tampoco si no se hubiera aumentado el poder de cómputo ni la creación de miles de dispositivos de “*Internet de las Cosas*”, que constantemente monitorean cientos de situaciones. Así que estamos en un punto de la Humanidad muy importante, en donde varias tecnologías han convergido para poder crear estas nuevas aplicaciones que están surgiendo día a día.

Actualmente las grandes superpotencias mundiales están en una batalla, la cual es por la supremacía en “*Inteligencia Artificial*”, en la cual, las naciones que se pongan adelante serán aquellas que tengan los mejores algoritmos, los mejores equipos de procesamiento y una gran cantidad de datos que hagan posible todas estas aplicaciones.

Dentro de todos estos cambios mundiales que están sucediendo en estos momentos es que propuse como proyecto de mi año sabático la creación de este libro, por ser un tema de gran actualidad y en donde a pesar de haber mucha información, la mayoría está dispersa y orientada a diferentes áreas y temas. Decidí este tema no por ser experto, si no precisamente por conocer mas de esta área, en donde afortunadamente conozco algunos de estos temas donde he impartido clases durante varios años (bases de datos, redes, programación, inteligencia artificial, internet de las cosas, etc.), los cuales me han permitido conjuntar mis conocimientos previos con la investigación que he realizado durante todo este año para la terminación de este libro.

Ha sido un reto hacer este libro, ya que sobre la misma marcha están sucediendo los cambios de tecnología a una gran velocidad, y varios temas y códigos he tenido que reescribirlos nuevamente por los cambios que han surgido. Nuevas tecnologías han surgido durante este último año y han puesto de moda todavía más la “*Ciencia de Datos*”, que mas que moda, es una tecnología basada en los datos, datos que siguen creciendo de manera exponencial. Este crecimiento demanda grandes cantidades de especialistas en esta área que por el momento no existen y que apenas los sistemas escolares de todo el mundo están empezando a implementar en sus retículas para cubrir la demanda actual y futura.

Este libro no fue pensado desde un principio para tocar a profundidad cada tema, ya que de cada unidad, se puede hacer un libro por completo (como los que existen actualmente). Este libro fue pensado como una guía para llevarte desde lo más básico hasta lo mas avanzado, y lograr entender las áreas que existen y la secuencia que hay que llevar a cabo para poder ir avanzando, para iniciar como principiante y terminar conociendo las herramientas mas avanzadas que existen en la actualidad sobre el área de “*Ciencia de Datos*”.

El siguiente paso es aplicar toda esta información en las clases y en las nuevas retículas que seguramente se modificarán para dar paso a la integración de todas estas nuevas tecnologías que ya demanda el mercado laboral actual.

Agradezco al “*Tecnológico Nacional de México*” y al “*Instituto Tecnológico de Morelia*” por haberme permitido la realización de mi “*Año Sabático*” realizando este libro de “*Ciencia de Datos, Teoría y Aplicaciones*”, ya que ha sido como un increíble viaje por la mundo de la tecnología actual, que culmina con la terminación de este libro, y mas que que un trabajo, ha sido un deleite haberlo realizado. ¡Muchas Gracias!

El futuro viene con grandes e increíbles tecnologías por delante, pero también con grandes retos a nivel mundial, por todos los cambios e impactos que tendrá el mundo con el uso de las nuevas tecnologías, y como todo en la vida, con cosas a favor y en contra. Y como conclusión me quedo con la siguiente frase de Yuval Noah Harari, de su libro “*21 lecciones para el siglo XXI*”, la cual sintetiza lo que viene por delante y la clara importancia del uso de los datos en los años por venir....

“*Quienes poseen los datos, poseen el Futuro!*”

Bibliografía

- (Amazon_01, 2022) Amazon. (2022). Machine Learning en AWS. <https://aws.amazon.com/es/machine-learning/>
- (Amazon_02, 2022) Amazon. (2022). Amazon Rekognition. <https://aws.amazon.com/es/rekognition/>
- (Ananthaswamy, 2021) Ananthaswamy, A. (2021, noviembre). La IA nos ha hecho vivir en un mundo basado en las probabilidades. MIT Technology Review. <https://www.technologyreview.es//s/13774/la-ia-nos-ha-hecho-vivir-en-un-mundo-basado-en-las-probabilidades>
- (APA, 2020) American Psychological Association. (2020). Publication manual of the American Psychological Association (7th ed.). American Psychological Association. <https://doi.org/10.1037/0000165-000>
- (Boudreau, 2020) Boudreau, E. (2020, diciembre). What Language Should You Learn for Data Science in 2021?. Towards Data Science. <https://towardsdatascience.com/what-language-should-you-learn-for-data-science-in-2021-fdeebb88d6e>
- (Brown et al, 2020) Brown, T. et al. (2020, july). Language Models are Few-Shot Learners. Computation and Language, Cornell University. <https://arxiv.org/pdf/2005.14165.pdf>
- (Cisco, 2021) Cisco. (2021). Network Topology Icons. Cisco. <https://www.cisco.com/c/en/us/about/brand-center/network-topology-icons.html>
- (Codd, 1970) Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>
- (Conacyt, 2022) Conacyt. (2022). COVID-19 Tablero México. CONACYT - CentroGeo. <https://datos.covid-19.conacyt.mx/>
- (Davenport, 2012) Davenport, T. (2012, octubre). Data Scientist: The Sexiest Job of the 21st Century. Harvard Business Review. <https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century>
- (Davis, 2021) Davis, A. (2021). Bootstrapping Microservices with Docker, Kubernetes and Terraform. Manning Publications
- (Deitel, 2020) Deitel, P. & Deitel. H. (2020). Intro to Python for Computer Science and Data Science. First Edition Pearson Education
- (Foxwell, 2020) Foxwell, H. (2020). Creating Good Data. Apress Publisher
- (Google_01, 2022) Google. (2022). Guía para principiantes de AutoML. <https://cloud.google.com/vertex-ai/docs/beginner/beginners-guide?hl=es>
- (Google_02, 2022) Google. (2022). Precios de Vertex AI. <https://cloud.google.com/vertex-ai/pricing>
- (HandWiki, 2021) HandWiki. (2021). HandWiki. <https://handwiki.org/wiki/>
- (Hao, 2020) Hao, K. (2020, noviembre). AI pioneer Geoff Hinton: "Deep learning is going to be able to do everything". MIT Technology Review. <https://www.technologyreview.com/2020/11/03/1011616/ai-godfather-geoffrey-hinton-deep-learning-will-do-everything/>
- (Harrison, 2015) Harrison, G. (2015). Next Generacion Databases. Apress Media USA

- (Hinton et al, 2006) First Edition
Hinton, G., Osindero, S. & Teh, Y. (2006). A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, Massachusetts Institute of Technology. <https://www.cs.toronto.edu/~hinton/absps/ncfast.pdf>
- (IEEE CS 1, 2022) IEEE Computer Society. (2022, abril). Computer Pioneers: Arthur Lee Samuel. <https://history.computer.org/pioneers/samuel.html>
- (ITM, 2021) Tecnológico Nacional de México Campus Morelia. (2021). Imagen Institucional. Tecnológico Nacional de México Campus Morelia. <https://sites.google.com/itmorelia.edu.mx/manualdeimagenalexterior/inicio>
- (Jaynes, 2003) Jaynes, E. (2003). Probability Theory: The Logic of Science. Cambridge University Press
- (Jenik, 2021) Jenike, C. (2021, julio). A Minute on the Internet in 2021. Statista. <https://www.statista.com/chart/25443/estimated-amount-of-data-created-on-the-internet-in-one-minute/>
- (JRBrown, 2022) JRBrown. (2022). Comparison standard deviations. Wikimedia. https://commons.wikimedia.org/wiki/File:Comparison_standard_deviations.svg
- (Kolmogorov, 1933) Kolmogorov, A. (1933). Foundations of the Theory of Probability. Internet Archive. <https://web.archive.org/web/20180914120850/http://www.mathematik.com/Kolmogorov/index.html>
- (Lighthill, 1972) Lighthill, J. (1972, julio). Artificial Intelligence: A General Survey. http://www.chilton-computing.org.uk/inf/literature/reports/lighthill_report/p001.htm
- (Loukides, 2021) Loukides, M. (2021, septiembre). 2021 Data/AI Salary Survey. O'Reilly. <https://www.oreilly.com/radar/2021-data-ai-salary-survey/>
- (Microsoft_01, 2022) Microsoft. (2022). Azure Machine Learning. <https://azure.microsoft.com/en-us/services/machine-learning/>
- (Microsoft_02, 2022) Microsoft. (2022). Azure Machine Learning pricing. <https://azure.microsoft.com/en-us/pricing/details/machine-learning/>
- (Mitchell, 1997) Mitchell, Tom. (1997). Machine Learning. McGrawHill. <http://www.cs.cmu.edu/~tom/mlbook.html>
- (Nagle et al., 2017) Nagle, T., Redman, T. & Sammon, D. (2017, september). Only 3% of Companies' Data Meets Basic Quality Standards. Harvard Business Review
- (Osmanovic, 2022) Osmanovic, A. (2022, june). We Asked GPT-3 to Write an Academic Paper about Itself—Then We Tried to Get It Published. Scientific American. <https://www.scientificamerican.com/article/we-asked-gpt-3-to-write-an-academic-paper-about-itself-then-we-tried-to-get-it-published/>
- (Patel et al., 2021) Patel, A. & Uppili, A. (2021, junio). Applied Natural Language Processing in the Enterprise. O'Reilly Media Sebastopol, CA USA First Edition
- (Raya, 2010) Raya, J. (2010). Guía de campo Máquinas Virtuales. Primera Edición Alfaomega Grupo Editor
- (Rielen et al., 2016) Rielen, D., Meysman, A. & Ali, M. (2016). Introducing Data Science". Manning Publications
- (Saharia et al., 2022) Saharia, c. et al. (2022, may). Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. Cornell University. <https://arxiv.org/pdf/2205.11487.pdf>
- (Spark, 2022) Spark. (2022). El clima y el tiempo promedio en todo el año en Morelia.

- (Stable_01, 2022) Weather Spark. <https://es.weatherspark.com/y/4452/Clima-promedio-en-Morelia-México-durante-todo-el-año>
- (Stable_02, 2022) Stable Diffusion. (2022). Stable Diffusion Public Release. <https://stability.ai/blog/stable-diffusion-public-release>
- (Stack_0001, 2021) Stable Diffusion. (2022). Stable Diffusion Demo. <https://huggingface.co/spaces/stabilityai/stable-diffusion>
- (Stanley, 2021) Stack Overflow. (2021). Developer Survey 2021. Stack Overflow. <https://insights.stackoverflow.com/survey/2021>
- (Stanley, 2021) Stanley, H. (2021). Introduction to Probability for Data Science. Michigan Publishing USA
- (Sucar, 2015) Sucar, L. (2015). Probabilistic Graphical Models. Springer-Verlag London First Edition
- (Sucar, 2022) Sucar, L. (2022). Course "Probabilistic Graphical Models: Principles and Applications". INAOE México. <https://ccc.inaoep.mx/~esucar/Clases-mgp/mgp.html>
- (Tamer, 2020) Tamer, M. & Valduriez P. (2020). Principles of Distributed Database Systems. Springer Nature Switzerland Fourth Edition.
- (TecNM, 2021) Tecnológico Nacional de México. (2021). Manual de Identidad Gráfica. Tecnológico Nacional de México. https://www.tecnm.mx/normateca/Direcci%C3%B3n%20de%20Cooperaci%C3%B3n%20y%20Difusi%C3%B3n/Manual_de_Identidad_Grafica_TecNM_2021.pdf
- (Tiku, 2022) Tiki, N. (11 de junio 2022). The Google engineer who thinks the company's AI has come to life. The Washington Post. <https://www.washingtonpost.com/technology/2022/06/11/google-ai-lambda-blake-lemoine/>
- (Toews, 2022) Toews, M. (2022). Standard deviation diagram. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Standard_deviation_diagram.svg
- (Torres, 2020) Torres, J. (abril 2020). Python Deep Learning. Alfaomega Grupo Editor, Ciudad de México, Primera Edición.
- (Torres, 2021) Torres, J. (mayo 2021). Introducción al aprendizaje por refuerzo profundo. Watch This Space Book Series, Barcelona España, Primera Edición.
- (Twitter, 2021) Twitter. (2021). Twemoji. Twitter. <https://twemoji.twitter.com/>
- (Vajjala et al., 2020) Vajjala, S, Majumder, B., Gupta, A & Surana, H. (2020). Practical Natural Language Processing. O'Reilly Media First Edition
- (Vaswani et al., 2017) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. & Polosukhin, I. (2017). Attention Is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
- (Vázquez, 2018) Vázquez, F. (2018, mayo). Creando inteligencia con Ciencia de Datos. Ciencia & Datos <https://medium.com/datos-y-ciencia/creando-inteligencia-con-ciencia-de-datos-d77be52c029e>
- (Vázquez, 2020) Vázquez, F. (2020, abril). Ciencia de Datos: Más que una tendencia. Talent Republic TV. <https://www.youtube.com/watch?v=Oo-i7AYx6q4>
- (Wiki_0001, 2021) Wikipedia. (2021). Ciencia de datos. Wikipedia. https://es.wikipedia.org/wiki/Ciencia_de_datos
- (Wiki_0002, 2021) Wikipedia. (2021). John W. Tukey. Wikipedia. https://es.wikipedia.org/wiki/John_W._Tukey
- (Wiki_0003, 2021) Wikipedia. (2021). John Graunt. Wikipedia.

- (Wiki_0004, 2021) https://es.wikipedia.org/wiki/John_Graunt
Wikipedia. (2021). Herman Hollerith. Wikipedia.
- (Wiki_0005, 2021) https://es.wikipedia.org/wiki/Herman_Hollerith
Wikipedia. (2021). Leo Breiman. Wikipedia.
- (Wiki_0006, 2021) https://en.wikipedia.org/wiki/Leo_Breiman
Wikipedia. (2021). Conjunto de datos. Wikipedia.
- (Wiki_0007, 2021) https://es.wikipedia.org/wiki/Conjunto_de_datos
Wikipedia. (2021). Archivo de texto. Wikipedia.
- (Wiki_0008, 2021) https://es.wikipedia.org/wiki/Archivo_de_texto
Wikipedia. (2021). Valores separados por comas. Wikipedia.
- (Wiki_0009, 2021) https://es.wikipedia.org/wiki/Valores_separados_por_comas
Wikipedia. (2021). PDF. Wikipedia.
- (Wiki_0010, 2021) <https://es.wikipedia.org/wiki/PDF>
Wikipedia. (2021). Extensible Markup Language. Wikipedia.
- (Wiki_0011, 2021) https://es.wikipedia.org/wiki/Extensible_Markup_Language
Wikipedia. (2021). JSON. Wikipedia.
- (Wiki_0012, 2021) <https://es.wikipedia.org/wiki/JSON>
Wikipedia. (2021). HTML. Wikipedia.
- (Wiki_0013, 2021) <https://es.wikipedia.org/wiki/HTML>
Wikipedia. (2021). Base de datos. Wikipedia.
- (Wiki_0014, 2022) https://es.wikipedia.org/wiki/Base_de_datos
Wikipedia. (2022). Computación en la nube. Wikipedia.
- (Wiki_0015, 2022) https://es.wikipedia.org/wiki/Computación_en_la_nube
Wikipedia. (2022). Red bayesiana.
- (Wiki_0016, 2022) https://es.wikipedia.org/wiki/Red_bayesiana
Wikipedia. (2022). Machine Learning.
- (Wiki_0017, 2022) https://en.wikipedia.org/wiki/Machine_learning
Wikipedia. (2022). Decision tree learning.
- (Wikimedia, 2021) https://en.wikipedia.org/wiki/Decision_tree_learning
Wikimedia. (2021). Wikimedia Commons.
- https://commons.wikimedia.org/wiki/Main_Page

Anexo

Tecnologías utilizadas

1

A continuación se muestran las tecnologías utilizadas en este libro:



Ciencia de Datos – Mapa de Tecnologías 2022

Tecnológico Nacional de México – Campus Morelia
Departamento de Sistemas y Computación



Python



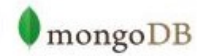
VSC



XLRD



OpenMPI



Matplotlib

Anexo

Instalación de Python

2

Instalación de Python en Linux (Ubuntu)

La mayoría de las distribuciones Linux ya traen instalado Python por default, en este caso se utilizó la distribución Ubuntu. Para revisar si ya está disponible Python abrimos una terminal y escribimos el comando que nos imprimirá la versión de Python (si es que está instalado):

```
$ python3 --version
```

Y en la terminal aparecerá:

```
Python 3.9.7
```

Si no está actualizado Python entonces actualizaremos los paquetes del sistema, para lo cual nos cambiamos al usuario administrador del sistema (root):

```
$ sudo su root
```

Nos pregunta nuestra contraseña y la escribimos (no aparece en pantalla ningún caracter de nuestra contraseña):

```
[sudo] contraseña para rogelio:
```

Si está todo correcto el indicador del sistema cambia a lo siguiente:

```
#
```

Para actualizar (update) los paquetes ya instalados, se hace con la siguiente instrucción:

```
# apt-get update
```

En este paso, la computadora se conectará a Internet y por lo tanto puede tardar un poco mientras verifica los paquetes instalados y cuáles descargará para actualizarlos.

También se recomienda verificar las nuevas versiones disponibles (upgrade) de los paquetes ya instalados, esto se hace con la siguiente instrucción:

```
# apt-get upgrade
```

En este paso, nuevamente la computadora se conectará a Internet y por lo tanto puede tardar un poco.

Nota: Se recomienda ejecutar los 2 pasos anteriores varias veces, hasta que marque que ya están actualizados todos los paquetes del sistema.

Nuevamente revisamos la versión de Python que ya está instalado:

```
$ python3 --version
```

Y nuevamente en la terminal aparecerá lo siguiente (la versión impresa en pantalla cambiará si hubo una actualización):

```
Python 3.9.7
```

Instalación de Python en Mac

Ir a la página oficial de Python:

<https://www.python.org/downloads/>

Descargar la última versión disponible (3.10.0 octubre 2021). El archivo que se descarga es:
python-3.10.0post2-macos11.pkg

Para iniciar la instalación se hace doble click sobre el archivo que acabamos de descargar y aparece una ventana con el "Instalador de Python", hacemos click en:

Continuar

Aparece otra pantalla con algunas instrucciones sobre esta versión de Python y hacemos click en:

Continuar

A continuación viene la licencia de uso del software y hacemos click en:

Continuar

Aparece una ventana encima donde haremos click para aceptar las condiciones de la licencia y hacemos click en:

Acepto

Después nos pregunta en qué unidad de disco se instalará (se requieren 152.4 MB), por default aparecerá el disco principal sobre el que estamos trabajando y hacemos click en:

Continuar

Aparece la siguiente pantalla donde nos indica el espacio que ocupará y el lugar donde se instalará (solo para confirmar las opciones previamente seleccionadas) y hacemos click en:

Instalar

Como es un software nuevo aparece una ventana donde nos pregunta nuestra clave de usuario administrador de nuestra computadora, lo tecleamos y hacemos click en:

Instalar software

En este momento se instala Python (dura poco tiempo) y al terminar la siguiente pantalla nos dice:
"Congratulations! Python 3.10.0 for macOS 10.9 or later was successfully installed."

Y hacemos click en:

Cerrar

Por último nos pregunta si el paquete original que descargamos (PKG) lo queremos conservar o enviar al basurero (papelera de reciclaje), si queremos conservarlo hacemos click en:

Conservar

Anexo

Instalación de Visual Studio Code

3

Instalación de Visual Studio Code (Linux)

Para instalar el Visual Studio Code (VSC) utilizaremos la distribución Ubuntu. Primero nos cambiamos al usuario administrador del sistema (root):

```
$ sudo su root
```

Nos pregunta nuestra contraseña y la escribimos (no aparece en pantalla ningún carácter de nuestra contraseña):

```
[sudo] contraseña para rogelio:
```

Si está todo correcto el indicador del sistema cambia a lo siguiente:

```
#
```

Ahora instalaremos 3 paquetes adicionales para poder descargar el VSC:

```
# apt-get install software-properties-common apt-transport-https wget
```

Ahora se importará la clave GPG de Microsoft usando "wget" y tecleamos en la consola lo siguiente:

```
# wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
```

El siguiente paso es habilitar el repositorio de VSC:

```
# add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
```

Ahora procederemos instalar VSC:

```
# apt-get install code
```

Ahora hay que salirse del modo administrador (root):

```
# exit
```

Y ya podremos ejecutar VSC escribiendo en la consola:

```
$ code
```

En pantalla y en modo gráfico aparecerá el VSC para poder empezar a trabajar con él.

Anexo

Instalación de bibliotecas en Python

4

Las Bibliotecas se encuentran en orden alfabético:

Actualizaciones

PIP

Para actualizar el "Sistema de Gestión de Paquetes de Python" (como conocido como PIP).

Sitio oficial: <https://pypi.org/project/pip/>

Para su actualización, ejecutar en la terminal:

```
> pip3 install --upgrade pip
```

Análisis Numérico

Numpy

Manejo de arreglos y otras operaciones matemáticas.

Sitio oficial: <https://pypi.org/project/numpy/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install numpy
```

Scipy

Es un biblioteca que depende de Numpy y tiene funciones para ser usadas en Matemáticas, especialmente en integración numérica y optimización.

Sitio oficial: <https://pypi.org/project/scipy/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install scipy
```

Pandas

Carga en memoria grandes cantidades de información para su mejor procesamiento.

Sitio oficial: <https://pypi.org/project/pandas/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install pandas
```

Configuraciones

nest-asyncio

Permite ejecutar y procesar eventos de manera asíncrona.

Sitio oficial: <https://pypi.org/project/nest-asyncio/>

Para su actualización, ejecutar en la terminal:

```
> pip3 install nest-asyncio
```

Bases de Datos

Covid

Conectar Python con el repositorio oficial de la Universidad Johns Hopkins y obtener estadísticas sobre el Covid.

Sitio oficial: <https://pypi.org/project/covid/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install covid
```

MYSQL-CONNECTOR-PYTHON

Instalar el ODBC para conectar Python con MySQL

Sitio oficial: <https://pypi.org/project/mysql-connector-python/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install mysql-connector-python
```

PYMONGO

Conectar Python con MongoDB para hacer transacciones.

Sitio oficial: <https://pypi.org/project/pymongo/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install pymongo
```

Cómputo Paralelo

MPI4PY

Conecta Python con el estándar MPI para implementar cómputo paralelo.

Sitio oficial: <https://pypi.org/project/mmpi4py/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install mpi4py
```

Fecha y Hora

PYTZ

Permite realizar cálculos de fecha y hora.

Sitio oficial: <https://pypi.org/project/pytz/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install pytz
```

Graficación

Matplotlib

Sirve para graficar datos

Sitio oficial: <https://pypi.org/project/matplotlib/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install matplotlib
```

Seaborn

Sirve para graficar datos estadísticos

Sitio oficial: <https://pypi.org/project/seaborn/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install seaborn
```

Machine Learning

Scikit-Learn

Sirve para procesar datos utilizando Machine Learning y Deep Learning

Sitio oficial: <https://pypi.org/project/scikit-learn/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install scikit-learn
```

TensorFlow

Sirve para procesar datos utilizando Machine Learning y Deep Learning

Sitio oficial: <https://pypi.org/project/tensorflow/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install tensorflow
```

Machine Learning (Herramientas en la Nube)

IBM Watson Machine Learning

Conjunto de herramientas de Machine Learning desarrolladas por IBM

Sitio oficial: <https://pypi.org/project/ibm-watson-machine-learning/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install ibm-watson-machine-learning
```

Manejo de Archivos

EXCEL

xlrd

Sirve para leer archivos que están en formato de Excel

Sitio oficial: <https://pypi.org/project/xlrd/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install xlrd
```

PDF

camelot

Sirve para leer tablas dentro de archivos PDF

Sitio oficial: <https://pypi.org/project/tabula-py/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install "camelot-py[cv]"
```

PyPDF2

Sirve para leer archivos que están en formato PDF

Sitio oficial: <https://pypi.org/project/PyPDF2/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install PyPDF2
```

tabula

Sirve para leer tablas dentro de archivos PDF

Sitio oficial: <https://pypi.org/project/tabula-py/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install tabula-py
```

XML

xlrd

Sirve para leer y procesar archivos que están en formato XML

Sitio oficial: <https://pypi.org/project/lxml/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install lxml
```

Procesamiento de Lenguaje Natural

NLTK

El "Toolkit para Lenguaje Natural" ó "Natural Language Toolkit", mas conocido como NLTK (por sus siglas en inglés) es una biblioteca especializada en el procesamiento del lenguaje natural.

Sitio oficial: <https://pypi.org/project/nltk/>

Para su instalación, ejecutar en la terminal:

```
> pip3 install nltk
```

StopWords

Contiene un conjunto de "Stop Words" (palabras vacías o sin significado) en varios idiomas para hacer segmentación de textos ("Tokenización").

Sitio oficial: <https://pypi.org/project/stop-words/>

Para su instalación, ejecutar en la terminal:
> pip3 install stop-words

TextBlob

Sirve para procesamiento de textos como clasificación, traducción y análisis de sentimientos.

Sitio oficial: <https://pypi.org/project/textblob/>

Para su instalación, ejecutar en la terminal:
> pip3 install textblob

Wordcloud

Sirve para generar una "nube de ideas" a partir de un conjunto de palabras.

Sitio oficial: <https://pypi.org/project/wordcloud/>

Para su instalación, ejecutar en la terminal:
> pip3 install wordcloud

Web Scraping

BeautifulSoup

Extrae información de una página Web (Web Scraping)

Sitio oficial: <https://pypi.org/project/beautifulsoup4/>

Para su instalación, ejecutar en la terminal:
> pip3 install beautifulsoup4

Tweepy

Extrae información de Twitter por medio de la API Oficial

Sitio oficial: <https://pypi.org/project/tweepy/1.7.1/>

Para su instalación, ejecutar en la terminal:
> pip3 install tweepy

Tweepy

Extrae información de Twitter

Sitio oficial: <https://pypi.org/project/twint/>

Para su instalación, ejecutar en la terminal:
> pip3 install twint

Wikipedia

Extrae y procesa información de Wikipedia

Sitio oficial: <https://pypi.org/project/wikipedia/>

Para su instalación, ejecutar en la terminal:
> pip3 install wikipedia

Anexo

Instalación de XAMPP

5

Instalación de Apache-PHP-MySQL en Linux

Existen varios programas que instalan Apache-PHP-MySQL, para esta instalación se eligió XAMPP el cual se descarga de la siguiente dirección:

<https://www.apachefriends.org/index.html>

El software se llama XAMPP y la última versión es la 8 (diciembre 2020). El archivo que se baja es: xampp-linux-x64-8.0.0-0-installer.run

Si no se han modificado las opciones de descarga del navegador, el archivo queda almacenado en "/home/rogelio/Descargas" (donde "rogelio" es el nombre del usuario de la máquina Linux). A continuación se arranca una terminal por medio de los menús superiores seleccionando: Aplicaciones > Accesorios > Terminal

Se cambia a usuario "root":
rogelio@rogelio-laptop:~\$ sudo su root
[sudo] password for rogelio:

Primero movemos el archivo que se bajó de internet ("xampp-linux-x64-8.0.0-0-installer.run") del directorio donde se descargó ("/home/rogelio/Descargas") al directorio donde lo vamos a instalar ("/opt"):
root@rogelio-laptop:/home/rogelio# mv /home/rogelio/Descargas/xampp-linux-x64-8.0.0-0-installer.run /opt/

Nos cambiamos al directorio /opt:
root@rogelio-laptop:/home/rogelio# cd /opt/

Le damos todos los permisos al archivo para poder instalarlo:
root@rogelio-laptop:/home/rogelio# chmod 777 xampp-linux-x64-8.0.0-0-installer.run

Ejecutamos el archivo para iniciar la instalación:
root@rogelio-laptop:/home/rogelio# ./xampp-linux-x64-8.0.0-0-installer.run

Aparece en una ventana que dice "Setup XAMPP" y hacemos click en:
Next

Aparece otra ventana que pregunta por los componentes que se van a instalar, lo dejamos como está y hacemos click en:
Next

Aparece otra ventana que nos indica que el "XAMPP" se va a instalar en el directorio:
/opt/lampp

Y hacemos click en:
Next

Aparece otra ventana donde nos pregunta si queremos saber mas acerca de otras herramientas de para XAMPP y deseleccionamos la casilla que inicialmente estaba marcada (no queremos información) y hacemos click en:
Next

Aparece otra ventana que nos dice que el XAMPP está listo para instalarse y hacemos click en:
Next

Luego inicia la instalación la cual tarda un poco y luego aparece una ventana donde nos indica que ya terminó y está marcada una casilla donde nos dice que arrancará el XAMPP y hacemos click en:
Finish

Regresa nuevamente al modo consola y ahora nos cambiamos al directorio que se acaba de crear:
root@rogelio-laptop:/opt# cd lampp/

y ahora se procede a arrancar el XAMPP con la siguiente instrucción:
root@rogelio-laptop:/opt/lampp# ./lampp start

y aparece en pantalla lo siguiente:
Starting XAMPP for Linux 8.0.0-0...
XAMPP: Starting Apache...
XAMPP: Starting MySQL...
XAMPP: Starting ProFTPD...
XAMPP for Linux started.

Para probar si funciona el servidor Web se arranca un navegador (en este caso Firefox) y se teclea:
http://localhost

Si está funcionando correctamente el servidor web, se muestra en la pantalla del navegador el logotipo de XAMPP en la parte central.

Si se apaga y arranca nuevamente la computadora hay que levantar nuevamente el servicio de Apache-PHP-MySQL, para ello primero arrancamos una terminal por medio de los menús superiores seleccionando:
Aplicaciones > Accesorios > Terminal

Luego nos cambiamos al usuario "root":
rogelio@rogelio-laptop:~\$ sudo su root
[sudo] password for rogelio:

Nos cambiamos al directorio /opt/lampp:
root@rogelio-laptop:/home/rogelio# cd /opt/lampp/

Y ahí arrancamos el XAMPP con la siguiente instrucción:
root@rogelio-laptop:/opt/lampp# ./lampp start

NOTA: Si se desea usar el phpMyAdmin (una página web para administrar el MySQL) se requiere

hacer click en el enlace de "phpMyAdmin" que se encuentra en la parte superior derecha de la página que aparece en el navegador cuando tecleamos "localhost".

Instalación de Apache-PHP-MySQL en Mac

Existen varios programas que instalan Apache-PHP-MySQL, para esta instalación se eligió XAMPP el cual se descarga de la siguiente dirección:

<http://www.apachefriends.org/en/xampp.html>

El software se llama XAMPP y la última versión es la 8.0.0 (diciembre 2020). El archivo que se baja es:

xampp-osx-8.0.0-0-vm.dmg

Hacer doble click encima del archivo y aparece una ventana con el ícono de XAMPP, el cual hay que hacer click y arrastrar hacia el ícono de la derecha (el ícono de Aplicaciones) y ahí lo soltamos.

Ahora nos cambiamos a la carpeta "Aplicaciones" donde se copió el XAMPP, y hacemos doble click encima para iniciar la instalación.

Sale una ventana que dice "XAMPP" es una aplicación descargada de Internet. ¿Seguro que quieres abrirla? y hacemos click en:

Abrir

Sale nuevamente una ventana que dice "This seems to be the first execution of the software" indicando que es la primera vez que se ejecutará y hacemos click en:

OK

Aparece una ventana que dice "stackman quiere realizar cambios", esto es para advertirnos que se harán cambios en el sistema, tecleamos nuestra clave de administrador de la computadora donde dice "Contraseña" y hacemos click en:

Ok

Aparece una ventana que dice "XAMPP" que es nuestro panel de administración del sistema, donde "Status" es de color amarillo y para iniciar el sistema hacemos click en:

Start

En la ventana se observa que "Status" cambia a color verde y la "IP Address" ya tiene una ip asignada, en este caso me asignó:

192.168.64.2

Por último para probar el funcionamiento del servidor y sus servicios, arrancamos un navegador Web (Firefox, Chrome, Opera, Safari, Edge, etc.) y tecleamos la dirección que nos asignó el sistema:

192.168.64.2

En el navegador se debe observar una página con la bienvenida al sistema:

XAMPP Apache + MariaDB + PHP + Perl

Welcome to XAMPP for 7.3.2-2

Indicando que el sistema ya se encuentra en funcionamiento.

Para crear una pagina hay que seleccionar la pestaña "Volumes" dentro de la ventana "XAMPP" y hacer click en:
Mount

Luego, dentro de la misma ventana se selecciona "Explore" y aparecerá una ventana con los archivos del XAMPP incluyendo la carpeta "htdocs" donde se crean las carpetas para crear nuevas páginas Web.

Esta nueva forma de trabajo es porque el XAMPP se monta como un sistema NFS, pero si se quiere ver la ruta dentro de la computadora donde se encuentra la carpeta "htdocs" se encuentra en:
/Users/rogelioferreira/.bitnami/stackman/machines/xampp/volumes/root/htdocs

Para que funcione el "phpMyAdmin" hay que seleccionar la pestaña "Network" dentro de la ventana "XAMPP" y hacer click en:
Enable

Con esto se habilita el acceso al "phpMyAdmin":
localhost:8080 -> 80 (Over SSH)

Para acceder al "phpMyAdmin" hay que abrir un navegador Web y teclear:
<http://localhost:8080/phpmyadmin>

Y en la pantalla aparecerá el "phpMyAdmin".

Anexo

Instalación de un Cluster

6

Instalación de un cluster con Raspberry Pi virtualizados

Pasos para la construcción del Cluster Virtualizado

- 1) Virtualizar Raspberry
- 2) Actualización y configuración de red.
- 3) Instalación de MPI4PY y SSH.

PASO 1) Virtualizar Raspberry

El nombre de la primer máquina será "Raspberry1" y será nuestro nodo "Maestro":

El nombre de la segunda máquina será "Raspberry2" y será nuestro nodo "Esclavo". En un cluster pequeño normalmente sólo hay un nodo "Maestro" y todos los demás serán "Esclavos". Se podrían agregar mas máquinas virtuales que funcionarán como "Esclavos" del nodo "Maestro".

PASO 2) Actualización y configuración de red

Para ambas máquinas virtuales hacer los siguientes pasos:

Cambio a usuario root:

```
$ sudo su root
```

Actualizar los paquetes del sistema:

```
$ apt-get update
```

Actualizar los cambios de versión de algunos paquetes del sistema:

```
$ apt-get upgrade
```

Ahora verificamos la dirección IP de cada máquina virtual:

```
$ ifconfig
```

Para este ejemplo:

La máquina "Maestro" es: 192.168.1.73

La máquina "Esclavo" es: 192.168.1.74

Nota: Como se puede observar ambas computadoras deben estar en el mismo segmento de red y deben de estar configuradas como de tipo "Puente", cuando se configura la máquina virtual, para que cada computadora tenga su propia IP y se comporte como una computadora independiente.

Ejecutamos un "ping" del Maestro al "Esclavo" para verificar la comunicación de red:
\$ ping 192.168.1.74

Verificamos la comunicación de red (Nodo Esclavo - Linux):

Y ejecutamos un "ping" ahora del "Esclavo" al "Maestro" para verificar la comunicación de red por los 2 lados:
\$ ping 192.168.1.73

PASO3) Instalación de MPI4PY y SSH

Para ambas máquinas virtuales hacer los siguientes pasos:

Seguimos trabajando en la consola como usuario "root" y ahora instalamos MPI4PY:
\$ apt-get install python3-mpi4py -y

Editamos el archivo hosts para colocar todas las computadoras del Cluster y escribimos:
\$ nano /etc/hosts

Nos abre el editor de texto y al final observamos nuestra computadora:
127.0.1.1 raspberry

Modificamos el nombre de nuestra computadora y esa línea debe quedar así: 127.0.1.1 raspberry1

Ahora le agregamos al final los datos de la computadora "Esclavo" (si estoy en la computadora "Maestro"):
192.168.1.74 raspberry2

Oprimimos CTRL + X para guardar el archivo y regresar a la consola.

Ahora hay que hacer lo mismo con la computadora "Esclavo" donde en el archivo "hosts" agregaremos al final:
192.168.1.73 raspberry1

También tenemos que cambiar el nombre de nuestra computadora, por default es "raspberrypi" y la cambiaremos a "raspberrypi1", ya que cada computadora del Cluster debe ser única, por lo cual editamos el archivo hostname:
\$ nano /etc/hostname

Nos abre el editor de texto y observamos el nombre actual de la computadora:
raspberrypi

Modificamos el nombre de nuestra computadora y esa línea debe quedar así:
raspberrypi1

Oprimimos CTRL + X para guardar el archivo y regresar a la consola.

Ahora hay que hacer lo mismo con la computadora "Esclavo" donde en el archivo "hostname" pondremos el nuevo nombre:
raspberrypi2

Nota: Para que el cambio de nombre surta efecto hay que reiniciar toda la máquina virtual en este momento, pero esto puede ocasionar un cambio de IP en alguna ó en ambas máquinas virtuales. Por lo anterior, en un cluster grande se ponen direcciones IP fijas a cada computadora. También se recomienda como primer paso de toda la instalación cambiar el nombre y reiniciar para que nuestras IPs no se afecten. En caso de haber algún cambio de IP se tiene que revisar y actualizar nuevamente el archivo "hosts".

Continuamos y para ambas máquinas virtuales hacer los siguientes pasos:

Ahora vamos a crear un usuario "mpirun" con el que estaremos trabajando la parte de cómputo paralelo:

```
adduser mpiuser
```

Nos pregunta la clave y le ponemos su mismo login (se puede poner cualquier clave que uno quiera):
mpiuser

Para confirmar nos vuelve a preguntar la clave y le ponemos nuevamente su mismo login:
mpiuser

Nos pregunta algunos otros datos (nombre, número, etc), a todos dar:
ENTER

Nos pregunta si es correcta la información y le escribimos:
S

Le damos ENTER y se termina la creación de la nueva cuenta de usuario (mpiuser) y nos regresa a la consola nuevamente.

La comunicación entre computadoras será por medio de SSH, por lo cual, lo primero que tenemos que hacer es activar el servicio, y por consola entramos a la configuración del Raspberry Pi:
\$ raspi-config

Aparece varias opciones y escogemos la opción 5 por medio de las flechas del teclado:
5 SSH

Nos pregunta si queremos habilitar el servicio y seleccionamos:
Si

Aparece un mensaje en pantalla indicando que ya está listo el SSH:
The SSH server is enabled

Seleccionamos:
Aceptar

Nos regresa a la pantalla anterior y seleccionamos:
Finish

Nos regresa nuevamente a la consola y verificamos si está en funcionamiento el SSH:
\$ service ssh status

Y debe aparecer en pantalla:
Active: active(running)

Ahora nos cambiamos al nuevo usuario "mpiuser":
su mpiuser

Verificamos que funcione el mpi y el nombre de nuestra computadora y escribimos en la consola:
mpirun hostname

En el Raspberry Pi sale (si estamos en el "Maestro"):
raspberrypi1

Probamos el ssh con los nombres de usuario creados:

Desde el Nodo Maestro al Esclavo (Raspberry Pi):
\$ ssh mpiuser@192.168.1.74

La primera vez que nos conectamos nos pregunta si queremos guardar la llave RSA y le decimos que:
yes

Nos debe preguntar el password del usuario "mpiuser":
Password: ?

Le tecleamos el password que le asignamos ("mpiuser") y debemos de entrar a la computadora remota. Para salir escribimos:
\$ exit

Y con eso se termina la conexión remota. Este paso hay que repetirlo pero ahora en modo inverso, es decir, hay que hacer un SSH del "Esclavo" al "Maestro" para probar la conexión.

Ahora vamos a generar la llave de comunicación RSA (seguimos como usuario "mpiuser"), para ello ejecutamos el siguiente comando:
\$ssh-keygen -t rsa

Nos pregunta el nombre del archivo donde se guarda la llave, le damos:
ENTER

Nos pregunta por una frase para generar la llave, nuevamente le damos:
ENTER

Para confirmar nos pregunta nuevamente la frase y nuevamente le damos:
ENTER

Nos genera e imprime en pantalla el "id", la llave "Pública" y termina la generación de llaves.

Ahora vamos a copiar las llaves a la otra computadora:
ssh-copy-id 192.168.1.74

Y trata de conectarse vía SSH con la computadora "Esclavo" y nos pregunta el password del usuario "mpiuser":

mpiuser password: ?

Le tecleamos el password y entonces puede hacer la copia de la llave a la otra computadora. Nos deja en la terminal remota y nos salimos con:
exit

Para probar si funciona podemos cambiarnos nuevamente a usuario "mpiuser":
su mpiuser

Y ahora intentar conectarnos con la computadora "Esclavo"
ssh mpiuser@192.168.1.74x

Debe de entrar, funcionar y ya no preguntarnos el password, lo cual indica que esta todo correcto.

Este paso hay que repetirlo pero ahora en modo inverso, es decir, hay que crear las llaves en el "Esclavo" y copiarlas al "Maestro".

Por último, para probar que todo funciona ejecutaremos en consola la siguiente instrucción:
\$ mpiexec -n 2 --hosts 192.168.1.73,192.168.1.74 hostname

Donde le especificamos que utilizaremos 2 nodos (-n 2), luego se ponen las direcciones de nuestros 2 nodos ("Maestro" y "Esclavo") y después que impriman su nombre (hostname) y debe de salir lo siguiente:

Para probar un programa en Python con MPI haremos los siguientes pasos en la consola en el "Maestro":

Nos cambiamos al usuario "mpiuser":
\$ su mpiuser

Nos pregunta el password:
contraseña: ?

Tecleamos la contraseña, y damos:
ENTER

Abrimos el editor de textos (nano) para crear el hola mundo:
nano hola_mundo.py

Tecleamos en el editor el código fuente del programa:

```
from mpi4py import MPI
size = MPI.COMM_WORLD.Get_size()
rank = MPI.COMM_WORLD.Get_rank()
name = MPI.Get_processor_name()
print("\nHola Mundo con MPI")
print("El tamaño del Cluster es: " + str(size))
print("Mi rango es: " + str(rank))
print("Mi nombre es: " + str(name))
```

Oprimimos CTRL + X para guardar el archivo y regresar a la consola.

Los pasos anteriores deben hacerse de igual manera en la computadora "Esclavo".

Para ejecutar ahora el programa anterior, escribiremos las siguientes instrucciones en el "Maestro":
`$ mpiexec -n 2 --hosts 192.168.1.73,192.168.1.74 python hola_mundo.py`

Donde le especificamos que utilizaremos 2 nodos (-n 2), luego se ponen las direcciones de nuestros 2 nodos ("Maestro" y "Esclavo") y después que ejecute Python el programa "hola_mundo.py" y debe de salir lo siguiente:

Aunque en el nodo "Esclavo" no aparece nada, el nodo "Maestro" si lo mandó llamar para que ejecutara el código, es decir, que el programa se ejecutó en ambos nodos, por eso cada nodo manda el tamaño del cluster, su rango dentro del Cluster y el nombre de la computadora.

Nota: El programa a ejecutar deberá ser "IDÉNTICO" tanto en "Maestro" como en "Esclavo", el código fuente debe ubicarse en el "MISMO" directorio y ejecutarse con el "MISMO" usuario para que funcione correctamente.

Anexo

Instalación de MongoDB

7

Instalación de MongoDB (Linux)

Este es el sitio oficial de MongoDB:

<https://www.mongodb.com/>

Pasos para instalar la versión oficial de MongoDB (recomendada) en vez de la versión no oficial que proporciona Ubuntu. La instalación para los diferentes sistemas operativos se encuentra aquí:

<https://docs.mongodb.com/manual/installation/>

La instalación específica para Linux Ubuntu se encuentra aquí:

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>

Abrimos una terminal y nos cambiamos al usuario "root":

```
> sudo su root
```

Importamos la llave pública PHP de MongoDB con el siguiente comando

```
> wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
```

Instalamos el paquete "gnupg":

```
> apt-get install gnupg
```

Importamos nuevamente la llave:

```
> wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
```

Creamos un archivo para el repositorio local de MongoDB:

```
> echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
```

Actualizamos el sistema para tome los cambios:

```
> apt-get update
```

Procedemos a instalar MongoDB:

```
> apt-get install -y mongodb-org
```

Agregamos las siguientes líneas para que cuando se actualice el sistema, también se actualicen los archivos de MongoDB:

```
echo "mongodb-org hold" | sudo dpkg --set-selections
echo "mongodb-org-server hold" | sudo dpkg --set-selections
echo "mongodb-org-shell hold" | sudo dpkg --set-selections
echo "mongodb-org-mongos hold" | sudo dpkg --set-selections
echo "mongodb-org-tools hold" | sudo dpkg --set-selections
```

Algunos sistemas linux usan el sistema antiguo para el manejo de procesos System V (comando

"service") y los mas nuevos usan systemd (comando "systemctl"), para averiguar cual se está empleando debemos teclear el siguiente comando:

```
> ps --no-headers -o comm 1
```

Para las versiones nuevas de Ubuntu debe de aparecer en la consola lo siguiente:
systemd

Ya que identificamos cual comando debemos usar, ahora si procederemos a arrancar MongoDB:
> systemctl start mongod

No debe aparecer nada en la consola (a menos que haya errores). Si hay errores que se marcan en pantalla (" Failed to start mongod.service: Unit mongod.service not found."), se pueden volver a recargar los procesos con el siguiente comando:
> systemctl daemon-reload

Para verificar que está corriendo MongoDB tecleamos:
> systemctl status mongod

Deben de salir varios textos en pantalla incluyendo el siguiente mensaje (en color verde):
active (running)

Que indica que está corriendo nuestro MongoDB. Si tenemos algún problema podemos reiniciar MongoDB con la siguiente instrucción:
> systemctl restart mongod

Para entrar al "shell" de MongoDB y poder interactuar solamente tecleamos lo siguiente:
> mongo

Y debe de aparecer la siguiente pantalla del "shell" donde ya podemos interactuar directamente con MongoDB:

Para salir del "shell" y regresar a la consola, tecleamos el siguiente comando:
> exit

Ya en la consola, si queremos detener MongoDB:
> systemctl stop mongod

Instalación de MongoDB (Mac)

Se requiere instalar las herramientas para línea de comandos que tiene Xcode de Apple, para lo cual se abre una terminal y se teclaea lo siguiente:
> xcode-select --install

Se requiere instalar "Homebrew", y ésta es su página oficial:
Homebrew <https://brew.sh/>

Para instalar Homebrew, se escribe lo siguiente en la terminal (preguntará por la clave del administrador):
> /bin/bash -c "\$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Para poder instalar MongoDB requerimos instalar el "The MongoDB Homebrew Tap" escribiendo lo siguiente en consola (donde /opt/homebrew/bin es el directorio de homebrew, aunque en versiones antiguas de Mac está en /usr/local/Homebrew/bin):
`/opt/homebrew/bin/brew tap mongodb/brew`

Lo siguiente es instalar MongoDB escribiendo en consola:
`/opt/homebrew/bin/brew install mongodb-community@4.4`

Para arrancar MongoDB escribimos en consola:
`/opt/homebrew/bin/brew services start mongodb-community@4.4`

Y debe observarse en la consola:
==> Successfully started `mongodb-community` (label: homebrew.mxcl.mongodb-commu

Para detener MongoDB:
`/opt/homebrew/bin/brew services stop mongodb-community@4.4`

Y debe observarse en la consola:
==> Successfully stopped `mongodb-community` (label: homebrew.mxcl.mongodb-commu

Anexo

Instalación de una máquina virtual

8

Instalación de VirtualBox (Mac)

Primero hay que ir al sitio oficial de Oracle:

<https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html>

Y de ahí descargar los 2 archivos que son las últimas versiones de VirtualBox (hasta noviembre 2020):

Oracle VM VirtualBox 6.1.26: VirtualBox-6.1.26-145957-OSX.dmg

Oracle VM VirtualBox Extension Pack 6.1.26: Oracle_VM_VirtualBox_Extension_Pack-6.1.26-145957.vbox-extpack

A continuación se ejecuta el siguiente haciendo doble click encima del archivo de instalación: VirtualBox-6.1.26-145957-OSX.dmg

Aparece una ventana y hacemos click encima de:

VirtualBox.pkg

Después arrastramos el ícono a la carpeta "Applications" para que se copie el archivo de instalación (ya podemos cerrar la ventana que previamente nos había abierto).

Se abre carpeta "Aplicaciones" del sistema MacOS y aparece el archivo: VirtualBox.pkg

Hacemos doble click encima de él y aparece una ventana que nos pregunta si estamos seguros de continuar y verificar si el software es confiable, y se hace click en: Continuar

Luego viene una pantalla de bienvenida y se hace click en: Continuar

Luego sale una pantalla donde dice la ubicación donde se pondrá el VirtualBox (en este caso nos marca "Macintosh HD") y que la instalación nos llevará 253.8 MB de espacio en disco duro y hacemos click en: Instalar

Pregunta el nombre y contraseña del administrador de la computadora, lo tecleamos y hacemos click en:

Instalar

Inicia la instalación, y al terminar sale una ventana que dice "Instalación completada"

hacer click en:
Cerrar

Termina la instalación y cerramos la ventana.

NOTA: Se recomienda que antes de crear una máquina virtual se instale el "Oracle VM VirtualBox Extension Pack", el cual le agrega algunas funcionalidades a las máquinas virtuales que vayamos a instalar.

Para instalar el "Oracle VM VirtualBox Extension Pack" nos vamos nuevamente al sitio de Oracle:
<https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html>

Y de ahí descargamos el archivo:
Oracle_VM_VirtualBox_Extension_Pack-6.1.26-145957.vbox-extpack

Lo copiamos a nuestra carpeta "Aplicaciones" y encima de él hacemos doble click para ejecutarlo.

Aparece una ventana que nos indica que estamos a punto de instalar el "Extension Pack" y hacemos click en:
Instalar

Ahora aparecen los términos de la licencial del "Virtual Box" y si aceptamos los términos hacemos click en:
Acepto

Luego nos pregunta nuestra clave del usuario administrador de nuestra computadora para que pueda iniciar la instalación, le escribimos la clave y le damos "Enter" para continuar. Por último aparece una ventana que nos indica que "El paquete de Extension de Virtual Box se ha instalado correctamente" y hacemos click en:
Aceptar

Para crear nuestra primer máquina virtual abrimos la carpeta de Aplicaciones, y hacemos doble click sobre:
VirtualBox

A continuación se abre una ventana que dice "¡Bienvenido a VirtualBox!" y pasaremos a crear una máquina virtual

click en:
Nueva

Sale una pantalla y nos pregunta lo siguiente:
Nombre:
Tipo:
Versión:

En la parte de "Nombre" le ponemos el nombre tipo de sistema operativo a virtualizar Nombre:

Ubuntu_21-10

De manera automática se rellena "Tipo" con "Linux" y "Version" con "Ubuntu (64-bit)", y hacemos click en:

siguiente

Luego nos pregunta la cantidad de memoria base (RAM) asignada a la máquina virtual. El sistema recomienda 1024 MB pero si se tiene mas memoria es mejor asignar 2048. En este caso seleccionamos:

2048

y hacemos click en:

Siguiente:

Luego nos pregunta acerca del Disco Duro Virtual (el tamaño recomendado es de 10GB y así lo dejamos por el momento)

Seleccionamos

Crear un disco duro virtual ahora

Click en:

Crear

Ahora nos pregunta el "Tipo de archivo de unidad de disco duro":

seleccionamos

VDI (VirtualBox Disk Image) (Recomendado)

y hacemos click en:

Siguiente:

Luego nos pregunta acerca del tamaño del disco (nos recomienda usar "Reservado dinámicamente" pero nos puede agotar el espacio de nuestro disco duro) y seleccionamos:

Tamaño Fijo:

y hacemos click en:

Siguiente:

Luego nos pregunta el nombre del archivo del nuevo disco virtual y por default nos deja el nombre que habíamos tecleado:

Ubuntu_21-10

Y seleccionamos el tamaño del disco virtual en megabytes. Por default trae 10.00 GB, y lo cambiamos a 19.99 GB ya que el Ubuntu requiere como mínimo 8.6 GB, y hacemos click en:

Crear

Aparece una barra de tiempo donde nos dice que esta creando el disco que espere un momento y después aparece en nuestra ventana inicial de VirtualBox la máquina virtual que acabamos de crear

Hacemos click en el botón:

Iniciar

A continuación a parece un asistente de primera ejecución de VirtualBox

Click en:
Siguiete

Luego aparece una ventana para seleccionar el medio de instalación.
En esta ventana seleccionamos la imagen ISO del sistema operativo a virtualizar para que inicie la instalación

(en esta ocasión será el archivo para crear un Ubuntu 21.10):
ubuntu-21.10-desktop-amd64.iso

Click en:
Siguiete

A continuación se abre una ventana y se instala el sistema operativo designado (en este caso Linux Ubuntu 21.10) de acuerdo a las instrucciones de instalación de cada sistema operativo.

Constantes en mayúsculas

Mala práctica: `x = 9.8`

Buena práctica: `ACELERACION = 9.8`

Colocar la definición de las constantes después de los "imports". Si son muchas constantes usar un archivo externo para almacenarlas:

`constants.py`

Usar verbos para definir funciones y clases:

Mala práctica: `def numeros_primos()`

Buena práctica: `def calcular_numeros_primos()`

Para declarar que un miembro es protegido ó privado dentro de una una clase, y que sólo puede ser accesado desde adentro de una clase, se usa la siguiente convención:

Se usa un guión bajo "_" para declarar un miembro protegido

Se usa un doble guión bajo "__" para declarar un miembro privado

Ejemplo

```
_variable_nombre = 'value' # miembro protegido
```

```
__variable_nombre = 'value' # miembro privado
```

No usar `import *`

Esto hace que no se conozcan las entidades que se van a importar dentro del código, es preferible especificar las entidades a importar. En caso de que sean muchas entonces si se recomienda importar todas usando `import *`

Buena práctica:

```
import paquete.modulo as nombre_corto
```

```
from paquete.modulo import ClaseEjemplo
```

Acceso a los imports

```
instancia_1 = paquete.modulo.ClaseEjemplo()
```

```
instancia_2 = nombre_corto.ClaseEjemplo()
```

```
instancia_3 = ClaseEjemplo()
```

Usar formateadores de código o Linters para revisar nuestro código. Uno de los mas usado es "pylint", para instalarlo:

```
pip install pylint
```

para revisar un código:

```
pylint codigo.py
```

Otro formateador es "black" el cual convierte el archivo al estándar PEP8. Es de los más usado por las empresas mas grandes (Facebook, Dropbox, KeepTruckin, Mozilla, Quora, Duolingo, QuantumBlack, Tesla, etc.):

```
https://peps.python.org/pep-0008/
```

Para instalar black:

```
pip3 install black
```

Para formatear un documento:

```
black archivo.py
```

Anexo

Lista de Datasets

10

A continuación se muestra una lista en orden alfabético de sitios en Internet de donde se pueden descargar diferentes tipos de “Datasets” de manera abierta y gratuita para todo el público:

- OpenML: <https://www.openml.org/>
- Scikit-Learn (integrados en la biblioteca): <https://scikit-learn.org/stable/datasets.html>
- UC Irvine Machine Learning Repository <https://archive.ics.uci.edu/ml/index.php>
- Kaggle Datasets <https://www.kaggle.com/datasets>
- Data Portals A Comprehensive List of Open Data Portals from Around the World <https://dataportals.org/>
- Open Data Monitor <https://opendatamonitor.eu/frontend/web/index.php?r=dashboard%2Findex>
- Nasdaq Data Link <https://data.nasdaq.com/search>
- List of datasets for machine-learning research https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research
- Tableau - Free Public Data Sets For Analysis <https://www.tableau.com/learn/articles/free-public-data-sets>
- Awesome Public Datasets <https://github.com/awesomedata/awesome-public-datasets>
- The home of the U.S. Government’s open data <https://data.gov/> <https://resources.data.gov/>
- Bright Data <https://brightdata.com/products/datasets>
- Open Data CERN <http://opendata.cern.ch/>
- Yelp Open Dataset <https://www.yelp.com/dataset>
- Open Government Data (OGD) Platform India <https://data.gov.in/>
- Reddit Datasets (foro) - <https://www.reddit.com/r/datasets/>
- KDnuggets Data: Government, State, City, Local and Public <https://www.kdnuggets.com/datasets/government-local-public.html>
- Crunchbase Data - <https://data.crunchbase.com/docs/daily-csv-export> - <https://data.crunchbase.com/docs>
- United States Census Bureau - <https://www.census.gov/data/datasets.html>
- Google Public Data - <https://www.google.com/publicdata/directory?hl=es>
- Infochimps Data: Useful datasets from all over - <https://github.com/infochimps-data/infochimps-data>
- Linked Open Data - <https://datahub.io/collections/linked-open-data>
- Factual Global Places - <https://www.factual.com/data-set/global-places/>
- Datos de libre acceso del Banco Mundial - <https://datos.bancomundial.org/>

Anexo

Oficio de Sabático

11

A continuación se anexa el oficio de autorización del “Año Sabático”:



Dirección General

Ciudad de México 18 de agosto de 2021

AUTORIZACIÓN No. AS-2-090/2021

JOSÉ LUIS GIL VÁZQUEZ
DIRECTOR DEL INSTITUTO
TECNOLÓGICO DE MORELIA
PRESENTE

Con fundamento en el Reglamento Interior de Trabajo del Personal Docente de los Institutos Tecnológicos, en las Políticas Académicas Generales del Periodo Sabático y las bases de la Convocatoria correspondiente, el Tecnológico Nacional de México a través de la Comisión Central, ha revisado la solicitud del Año Sabático del **C. ROGELIO FERREIRA ESCUTIA** en el Programa para la Elaboración de Apuntes, Libros, Objetivos Educativos y Reactivos de Evaluación bajo la modalidad: **B.2 Libro: "Ciencia de Datos, Teoría y Aplicaciones"**, se tiene a bien emitir la siguiente:

AUTORIZACIÓN

Del programa presentado a partir del **31 de agosto de 2021** hasta el **30 de agosto de 2022**, quedando su seguimiento bajo la responsabilidad de la Dirección a su cargo, debiendo recibir y evaluar por parte de la Comisión Dictaminadora y la Academia correspondiente, el reporte intermedio y final de las actividades desarrolladas durante el ejercicio del periodo sabático autorizado al docente. Así mismo, se deberá remitir a la Secretaría Académica, de Investigación e Innovación a través de la Dirección de Docencia e Innovación Educativa del Tecnológico Nacional de México, la evaluación de dichos reportes que comprenderán la totalidad de lo establecido en el proyecto académico calendarizado (formato PS-02), que se encuentra en el expediente de la solicitud del docente; además de un ejemplar en físico y en electrónico (en formato PDF) del producto final comprometido.


ENRIQUE FERNÁNDEZ FASSNACHT
DIRECTOR GENERAL
TECNOLÓGICO NACIONAL DE MÉXICO


SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
DIRECCIÓN GENERAL



Av. Universidad 1200, Col. Xoco, Alcaldía Benito Juárez,
C.P. 03330, Ciudad de México.
Tel. (55) 3601-7500, Ext. 65050, e-mail: direccion@tecnm.mx
tecnm.mx





Anexo

Oficio de Sabático (Revisión)

12

A continuación se anexa el oficio de de Revisión Final del “Año Sabático”:

 **EDUCACIÓN**
SECRETARÍA DE EDUCACIÓN PÚBLICA


 **TECNOLÓGICO NACIONAL DE MÉXICO**

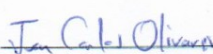
Instituto Tecnológico de Morelia
Subdirección Académica
Departamento de Sistemas y Computación


FORMATO CONCENTRADOR DE CEDULAS DE EVALUACIÓN DE LIBRO O APUNTES

DOCENTE:	Rogelio Ferreira Escutia					
ACADEMIA:	Departamento de Sistemas y Computación					
ASIGNATURA O DISCIPLINA:	Ingeniería en Sistemas Computacionales					

Criterios de Evaluación	Puntuación Máxima	Revisor 1	Revisor 2	Revisor 3	Observaciones	Calificación
I. Organización Temática	20	20	20	20		20
II. Desarrollo del tema	50	50	50	50		50
III. Dominio del tema	30	30	30	30		30
TOTAL	100%	PROMEDIO FINAL				100


REVISOR 1 


REVISOR 2 

REVISOR 3 


NOMBRE Y FIRMA Jesús Eduardo Alcaraz Chávez	NOMBRE Y FIRMA Juan Carlos Olivares Rojas	NOMBRE Y FIRMA Juan Jesús Ruíz Lagunas
---	---	--

LUGAR Y FECHA: Morelia Michoacán a 31 de agosto del 2022


 **EDUCACIÓN**
SECRETARÍA DE EDUCACIÓN PÚBLICA

 **TECNOLÓGICO NACIONAL DE MÉXICO**

INSTITUTO TECNOLÓGICO DE MORELIA
INGENIERÍA EN SISTEMAS COMPUTACIONALES



Av. Tecnológico #1500, Col. Lomas de Santiaguito, C.P. 58120, Morelia, Michoacán.
Tel. (443) 3121570 Ext. 233, e-mail: sistemas@morelia.tecnm.mx
tecnm.mx | morelia.tecnm.mx





ANÁLISIS TÉCNICO DE LA ACADEMIA

Fecha: 31 de agosto del 2022

C. José Luis Gil Vázquez
Director del Instituto Tecnológico de Morelia
Presente

Por medio de la presente, le comunicamos que habiendo sido revisado y evaluado el reporte de la persona cuyos datos se mencionan a continuación:

Docente:	Rogelio Ferreira Escutia		RFC:	FEER711125N34	
No. Autorización:	AS-2-090/2021		Periodo:	31-08-2021 al 30-08-2022	
Periodo reprogramado del: (Solo llenar en caso de haberlo solicitado)		Oficio Núm.		De fecha:	
Programa autorizado:	B.2	Libro			
	(a,b,c,d,e,)	(Programa general)			
"Ciencia de Datos, Teoría y Aplicaciones"		Instituto Tecnológico de Morelia			
(Título específico)		(Instituto o empresa)			
Cambio de programa autorizado: (Solo llenar en caso de haberlo solicitado)					
	(a,b,c,d,e,)	(Programa general)			
(Título específico)		(Instituto o empresa)			
Con oficio Núm.:	AS-2-090/2021		De fecha:	18-08-2021	
Informe:	Intermedio:	()	Final:	(X)	
Periodo reportado:	31-08-2021 al 30-08-2022		Que cubre:	100%	
				(50-100)	

de avance de las actividades del proyecto académico calendarizado (formato PS-02).

Esta Academia determina que **cumple técnica y académicamente** con lo establecido en el programa autorizado y en las Políticas Académicas Generales.

Atentamente

J. Guadalupe Ramos Díaz
Presidente de la Academia de Sistemas y Computación

Vo.Bo

José Martín Carlos Pérez
Jefe del Departamento de Desarrollo Académico

Jesús Eduardo Alcaraz Chavez
Profesor revisor Catedrático del área de Sistemas Computacionales

Juan Carlos Olivares Rojas
Profesor revisor Catedrático del área de Sistemas Computacionales

Juan Jesús Ruiz Lagunas
Profesor revisor Catedrático del área de Sistemas Computacionales

C.c.p. Subdirección Académica
Departamento de Desarrollo Académico
Departamento Académico
Comisión Dictaminadora e Interesado (a).



Av. Tecnológico #1500, Col. Lomas de Santiaguito, C.P. 58120, Morelia,
Tel. (443) 3121570 Ext. 214, e-mail: sub.academica@morelia.tecnm.mx
tecnm.mx | morelia.tecnm.mx



2022 Flores
Año de Magón
PRELUDIO DE LA REVOLUCIÓN MEXICANA