# Módulo 1. Python como lenguaje para desarrollar IA

CÓRDOBA CREA FUTURO

# Introducción

En uno de los cursos anteriores se hizo mención de los paradigmas de programación, indicando que un paradig estilo, método o filosofía fundamental para estructurar y resolver problemas de software. Dicta cómo pensar y organ lenguaje en sí, sino de un conjunto de reglas y principios (como POO, funcional o estructurado) que definen las her formas válidas de combinarlas para construir programas, ayudando a escribir código más eficiente y mantenible.

También se mencionaron distintos paradigmas, entre ellos, los siguientes:

- Paradigma orientado a la interpretación y a la compilación.
- Paradigma orientado a la programación de listas.
- Paradigma de programación en un contexto de IA.
- Paradigma de programación en un entorno de datos masivos.

A continuación, se presenta una tabla con algunos de los lenguajes de programación existentes y su clasificación que representan.

**Tabla 1. Paradigmas y lenguajes de programación**

| Paradigma | Lenguaje | Características | Ventajas | Desventajas | Año de creación | Autor |
|---|---|---|---|---|---|---|
| Cuántico | Q# | Diseñado para algoritmos cuánticos, con integración en Azure Quantum. | Optimizado para computación cuántica. | Uso limitado en *hardware* real actual. | 2017 | Microsoft |
| Cuántico / Funcional | Quipper | Diseñado para describir | Adecuado para computación | Uso limitado fuera del ámbito cuántico. | 2013 | Peter Selinger et al. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | algoritmos cuánticos. | cuántica a gran escala. | | | |
| Orientado a objetos | Swift | Lenguaje moderno de Apple para iOS y macOS. | Seguro, rápido y con sintaxis moderna. | Enfocado en el ecosistema Apple. | 2014 | Apple Inc |
| Orientado a objetos | Java | Encapsulamiento, herencia y polimorfismo. | Modularidad y reutilización de código. | Mayor complejidad que los lenguajes procedimentales. | 1995 | James Gosling |
| Orientado a objetos / Funcional | Kotlin | Moderno, conciso e interoperable con Java. | Seguro y fácil de mantener, ideal para desarrollo Android. | Comunidad menor que Java. | 2011 | JetBrains |
| Multiparadigma | Julia | Alto rendimiento, tipado dinámico, orientado a datos científicos. | Rápido, ideal para datos masivos y simulaciones científicas. | Ecosistema más joven que Python o R. | 2012 | Jeff Bezanson Stefan Karpinski, Viral B Shah, Alan Edelman |
| Multiparadigma | Rust | Seguridad de memoria sin recolector de basura. | Seguridad y rendimiento. | Curva de aprendizaje. | 2010 | Graydon Hoare |
| Multiparadigma | Go | Compilado, concurrente y eficiente. | Simplicidad y rendimiento, | Falta de genéricos (hasta versiones r | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | ideal para servidores. | | | |

Fuente: elaboración propia

Un lenguaje de programación muy usado en el mundo de la ciberseguridad y ciberdefensa con sólida ace profesionales es el lenguaje Python que soporta el trabajo de datos masivos, machine learning, etc.

Este lenguaje es el que adaptaremos en este capítulo para enseñanza de los temas propuestos.

CONTINUAR

# 1. Python

Con la premisa de que este es un curso avanzado, si no dispones de conocimientos previos o deseas familiarizarte con el lenguaje, se sugiere que consultes la primera referencia al final del presente apunte. Para incursionar en Python aplicado al *machine learning*, se recomienda que revises la segunda referencia incluida al final del documento.

Python es el estándar de la industria para desarrollar inteligencia artificial debido a su simplicidad sintáctica y a un ecosistema de herramientas muy amplio. Su diseño permite a los desarrolladores centrarse en la lógica de los algoritmos en lugar de en la complejidad del código.

Entre las razones que explican su amplia adopción en el campo de la inteligencia artificial, se encuentran las siguientes:

- **Curva de aprendizaje baja**. Su sintaxis es clara, legible y similar al pseudocódigo, lo que facilita la experimentación rápida y el prototipado.
- **Ecosistema amplio**: cuenta con bibliotecas especializadas que cubren todas las etapas de la IA, desde el preprocesamiento de datos hasta el despliegue de modelos.
- **Comunidad y soporte**: el respaldo de organizaciones como Google (TensorFlow) y Meta (PyTorch) garantiza actualizaciones constantes y abundante documentación.

Para trabajar en inteligencia artificial con Python, a continuación se presentan las herramientas fundamentales, clasificadas según su función:

**Tabla 2. Herramientas fundamentales de Python para inteligencia artificial**

| Categoría | Herramientas clave |
|---|---|
| **Aprendizaje automático (ML)** | Scikit-learn (clásico), XGBoost. |
| **Aprendizaje profundo (*deep* | TensorFlow |

| | |
|---|---|
| *learning*) | (escalabilidad), PyTorch (investigación), Keras. |
| **Procesamiento de lenguaje (NLP)** | Hugging Face Transformers, spaCy, NLTK. |
| **Análisis de datos y matemáticas** | NumPy (cálculo), Pandas (manipulación), SciPy. |
| **Visión por computadora** | OpenCV. |

Fuente: elaboración propia

En el ámbito de la inteligencia artificial, existen diversos casos de uso que demuestran su aplicación práctica en distintos sectores. Entre los más comunes se encuentran los siguientes.

En primer lugar, el desarrollo de chatbots y agentes inteligentes. A través de la creación de *pipelines* con *LangChain*, es posible conectar modelos de lenguaje con fuentes de datos externas, lo que permite construir asistentes capaces de responder consultas, automatizar procesos y brindar soporte en tiempo real.

Otro caso frecuente es el análisis predictivo. Mediante técnicas de aprendizaje automático, se pueden realizar pronósticos de ventas, detectar fraudes o identificar patrones de comportamiento en entornos empresariales. Estas aplicaciones permiten anticipar escenarios y optimizar la toma de decisiones.

**Por último, el reconocimiento de imágenes constituye un área de amplio crecimiento. Se aplica, por ejemplo, en el diagnóstico médico por imagen y en sistemas de vigilancia autónomos, donde los modelos son capaces de identificar objetos, anomalías o situaciones específicas a partir del procesamiento visual.**

## Scikit-learn

Scikit-learn (también conocido como sklearn) es una biblioteca de código abierto para aprendizaje automático (ML) que admite aprendizaje supervisado y no supervisado.

Proporciona un conjunto amplio de herramientas para tareas habituales de análisis y modelado de datos, con una interfaz intuitiva y consistente.

Se utiliza ampliamente tanto en la industria como en el ámbito académico, y existe una gran cantidad de tutoriales y fragmentos de código disponibles en línea.

Scikit-learn se basa en otras bibliotecas científicas de Python, como NumPy y SciPy, y se integra con Pandas y Matplotlib para el manejo y la visualización de datos.

Entre sus principales funcionalidades se encuentran las siguientes:

**Clasificación.** –

Identifica la categoría a la que pertenece un objeto (por ejemplo, detección de spam o identificación de enfermedades).

**Regresión:** –

predice un valor numérico continuo (por ejemplo, el precio de una vivienda en función de determinadas características).

**Agrupamiento:** –

agrupa objetos similares en datos sin etiquetar (por ejemplo, detección de anomalías en ciberseguridad).

**Reducción de la dimensionalidad:** –

reduce el número de variables consideradas para simplificar el análisis de datos (por ejemplo, en investigación genómica o análisis de texto).

**Selección y evaluación de modelos:** –

ofrece herramientas para evaluar el rendimiento mediante métricas como exactitud, precisión y puntuación $R^2$, además de técnicas como la validación cruzada y el ajuste de hiperparámetros.

**Preprocesamiento de datos:** –

incluye utilidades para la limpieza y preparación de datos, como la gestión de valores faltantes, el escalado de variables numéricas y la codificación de variables categóricas.

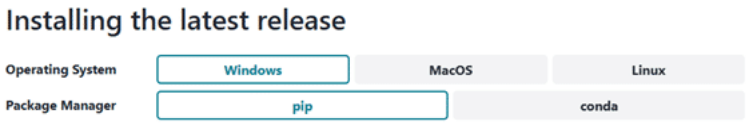
## ¿Por qué usar Scikit-learn?

En cuanto a las razones para utilizar esta biblioteca, pueden mencionarse las siguientes:

- **Facilidad de uso**. Ofrece una API sencilla y consistente en distintos modelos, lo que la convierte en una opción adecuada tanto para principiantes como para profesionales.
- **Eficiencia**: dispone de algoritmos prediseñados y optimizados para diversas tareas.
- **Uso profesional y académico**: se emplea en entornos profesionales de distintos sectores basados en datos, como finanzas, salud y tecnología, así como en investigación académica.
- **Código abierto**: disponible bajo la licencia BSD, es gratuito para uso comercial y cuenta con una comunidad amplia que mantiene actualizaciones periódicas y documentación extensa.

Para comenzar a utilizarla, puedes instalar scikit-learn mediante el comando `pip install scikit-learn` en tu entorno Python. El sitio web oficial ofrece tutoriales y guías que permiten aprender e implementar modelos de aprendizaje automático de manera adecuada.

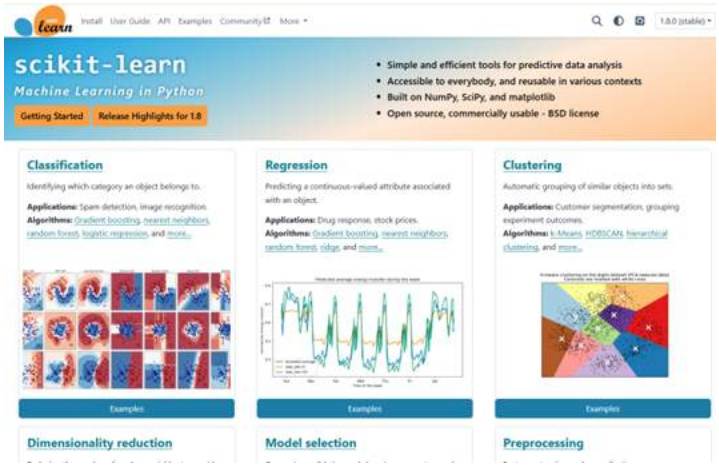
**Figura 1. Instalación de Scikit-learn**



Fuente: captura de pantalla de sitio web de Scikit-learn (https://goo.su/OBQL)

Al acceder al sitio web https://scikit-learn.org/stable/, podrás apreciar lo que se muestra en la siguiente imagen. Allí encontrarás modelos ya desarrollados que puedes utilizar para aprender, mediante la lectura de la documentación y la instalación de los recursos disponibles.

**Figura 2. Página principal del sitio web oficial de Scikit-learn**




Fuente: captura de pantalla de sitio web de Scikit-learn (https://goo.su/OBQL)

JUPYTER NOTEBOOK | NUMPY

Jupyter Notebook es un entorno interactivo que permite ejecutar código desde el navegador. Es una herramienta adecuada para el análisis exploratorio de datos y se utiliza ampliamente en el ámbito de la ciencia de datos. Aunque es compatible con diversos lenguajes de programación, en este curso solo utilizaremos su integración con Python.

JUPYTER NOTEBOOK | NUMPY

NumPy es uno de los paquetes fundamentales para la computación científica en Python. Incluye funcionalidades para trabajar con matrices multidimensionales, funciones matemáticas de alto nivel —como operaciones de álgebra lineal y la transformada de Fourier— y generadores de números pseudoaleatorios.

En Scikit-learn, la matriz de NumPy constituye la estructura de datos fundamental, ya que la biblioteca recibe la información en este formato. Por lo tanto, cualquier conjunto de datos que utilices deberá

convertirse en una matriz de NumPy.

La funcionalidad principal de NumPy se centra en la clase `ndarray`, que representa un arreglo multidimensional (*n*-dimensional). Todos los elementos del arreglo deben ser del mismo tipo de dato. Un ejemplo sencillo es el siguiente:

```
import numpy as np
x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))
```

Este código produce la siguiente salida:

```
x:
[[1 2 3]
 [4 5 6]]
```

## SciPy

SciPy es una colección de funciones para la computación científica en Python. Proporciona, entre otras funcionalidades, rutinas avanzadas de álgebra lineal, optimización de funciones matemáticas, procesamiento de señales, funciones matemáticas especiales y distribuciones estadísticas.

Scikit-learn se basa en funciones de SciPy para implementar sus algoritmos. La parte más relevante en este contexto es el módulo «scipy.sparse», que permite trabajar con matrices dispersas, otra representación de datos utilizada en Scikit-learn.

Las matrices dispersas se emplean cuando se necesita almacenar una matriz bidimensional que contiene principalmente valores cero.

A continuación, se muestra un ejemplo en el que se crea una matriz densa con NumPy:

```
from scipy import sparse

# Cree una matriz NumPy 2D con una diagonal de unos y ceros en el resto del espacio.

eye = np.eye(4)

print("NumPy array:\n{}".format(eye))
```

La ejecución del código anterior produce el siguiente resultado:

```
NumPy array:
```

```
[[ 1. 0. 0. 0.]

 [ 0. 1. 0. 0.]

 [ 0. 0. 1. 0.]

 [ 0. 0. 0. 1.]]
```

Posteriormente, la matriz puede convertirse en una matriz dispersa de SciPy en formato CSR, donde solo se almacenan los valores distintos de cero:

```
# Convierte la matriz NumPy en una matriz dispersa SciPy en formato CSR.

# Solo se almacenan las entradas distintas de cero.

sparse_matrix = sparse.csr_matrix(eye)

print("\nSciPy sparse CSR matrix:\n{}".format(sparse_matrix))
```

Al ejecutar este fragmento, se obtiene:

```
Matriz de RSE dispersa de SciPy:

 (0, 0) 1.0

 (1, 1) 1.0

 (2, 2) 1.0

 (3, 3) 1.0
```

En muchos casos no es posible crear primero una representación densa de los datos dispersos, ya que podría no caber en memoria. Por ello, es necesario generar directamente la matriz en formato disperso. A continuación, se muestra cómo crear la misma matriz utilizando el formato COO:

```
data = np.ones(4)

row_indices = np.arange(4)
```

```
col_indices = np.arange(4)

eye_coo = sparse.coo_matrix((data, (row_indices, col_indices)))

print("COO representacion:\n{}".format(eye_coo))
```

La salida correspondiente se muestra a continuación:

```
COO representacin:

 (0, 0) 1.0

 (1, 1) 1.0

 (2, 2) 1.0

 (3, 3) 1.0
```

## Pandas

Pandas es una biblioteca de Python para la manipulación y el análisis de datos. Se basa en una estructura denominada DataFrame, inspirada en el DataFrame de R. En términos simples, un DataFrame de Pandas es una tabla, similar a una hoja de cálculo de Excel.

Pandas ofrece numerosos métodos para modificar y operar con estas tablas; en particular, permite realizar consultas y uniones de datos de manera similar a SQL. A diferencia de NumPy, que requiere que todos los elementos de una matriz sean del mismo tipo, Pandas permite que cada columna tenga un tipo diferente (por ejemplo, enteros, fechas, números de punto flotante o cadenas de texto).

Otra herramienta destacada es su capacidad para importar y procesar datos desde diversos formatos de archivo y bases de datos, como SQL, archivos de Excel o archivos de valores separados por comas (CSV).

Detallar todas las funcionalidades de Pandas excede el alcance de este apunte. No obstante, el libro *Python for Data Analysis*, de Wes McKinney (O'Reilly, 2012), constituye una guía de

referencia.

A continuación, se muestra un ejemplo sencillo de creación de un DataFrame a partir de un diccionario:

```
import pandas as pd

# create a simple dataset of people

data = {'Name': ["John", "Anna", "Peter", "Linda"],

 'Location' : ["New York", "Paris", "Berlin", "London"],

 'Age' : [24, 13, 53, 33]

 }

data_pandas = pd.DataFrame(data)

# IPython.display permite la impresión elegante de marcos de datos

# en  Jupyter notebook

display(data_pandas)
```

La ejecución del código anterior genera una tabla con el siguiente formato:

**Tabla 3. Ejemplo de DataFrame generado con Pandas**

| | Age | Location | Name |
|---|---|---|---|
| **0** | 24 | New York | John |
| **1** | 13 | Paris | Anna |
| **2** | 53 | Berlin | Peter |
| **3** | 33 | London | Linda |

Fuente: elaboración propia

## Datos

Así como una computadora, o casi cualquier dispositivo, está formada por hardware y software, dentro de estos componentes existen subdivisiones necesarias. Hay hardware específico que, por su rendimiento, puede aprovecharse para determinadas tareas, como servidores, inteligencia artificial o uso doméstico.

Lo mismo ocurre cuando hacemos referencia al software: existen lenguajes de programación que se aplican a determinados paradigmas. Si profundizamos en los lenguajes de programación, encontramos, por un lado, la aplicación que gestiona y administra los datos. Este aspecto se transforma en un elemento central de todo sistema, por lo que modelizarlo y mantenerlo adquiere gran importancia en el ciclo de vida del ecosistema digital.

Esto puede ejemplificarse con facilidad. Este apunte se está desarrollando en un editor de texto, pero el valor reside en el contenido escrito. Por ello, se han tomado las precauciones necesarias para su conservación. El editor de texto puede descargarse nuevamente, activarse la licencia de uso y continuar utilizándose sin inconvenientes; en cambio, el archivo con el formato que ha acompañado toda la capacitación requiere especial cuidado.

En *machine learning*, los datos poseen la misma relevancia, con el añadido de que deben modelizarse adecuadamente para que el algoritmo pueda realizar predicciones de manera correcta. En este contexto, los datos se almacenan en un repositorio denominado ***dataset***.

## Dataset

Un *dataset* (conjunto de datos), en *machine learning*, es una colección organizada de información estructurada para entrenar, probar y evaluar modelos de inteligencia artificial. Actúa como materia prima y suele presentarse en formatos tabulares (filas y columnas) o en formatos multimedia, lo que permite a los algoritmos identificar patrones y realizar predicciones.

En cuanto a su **estructura**, normalmente se organiza en:

- filas, que representan entradas o muestras;
- columnas, que representan variables o características.

Respecto de su **contenido**, puede estar compuesto por datos numéricos, datos categóricos, texto, imágenes, videos o audios.

Según su finalidad dentro del modelo, los datos pueden clasificarse del siguiente modo:

- Datos de entrenamiento, utilizados para que el modelo aprenda.
- Datos de prueba (*testing*), empleados para evaluar la precisión del modelo.

Cabe señalar que un *dataset* no debe valorarse únicamente por su tamaño; su calidad depende de que los datos sean relevantes, representativos y estén libres de errores o sesgos.

Respecto a los **formatos**, los más comunes son .csv (valores separados por comas), .json y .zip, además de estructuras como los DataFrame.

Algunos ejemplos de *dataset* son los siguientes:

- **Iris dataset**, utilizado para clasificar especies de flores.
- Conjuntos de imágenes empleados en visión por computadora.
- Registros financieros utilizados para la detección de fraudes.

**Los dataset constituyen la base del aprendizaje automático, ya que los modelos de machine learning se alimentan de estos datos para aprender y generar predicciones. Por ejemplo, un dataset de ventas puede utilizarse para anticipar la demanda futura de un producto.**

**¿Había en el siglo XVIII IA en Argentina?** —

Con el propósito de aportar un ejemplo que permita comprender el concepto de *dataset* y la predicción mediante algoritmos de IA, puede recordarse un dicho popular en Argentina: «me duelen los huesos, entonces va a cambiar el clima». Esta expresión ha sido transmitida de generación en generación dentro de numerosas familias. En consecuencia, podría afirmarse que, al menos desde algún momento del siglo XIX, existían formas rudimentarias de «algoritmo» y «*dataset*» basadas en la experiencia acumulada y la observación empírica.

En la actualidad, existe respaldo científico que vincula el dolor articular —percibido como «dolor de huesos»— con los cambios climáticos, afectando especialmente a personas con artrosis o artritis. La disminución de la presión atmosférica antes de la lluvia puede expandir los tejidos, mientras que el frío incrementa la rigidez articular y el dolor (Farmalastic, s. f.).

Sabiendo que hay personas sensibles a los cambios meteorológicos, nuestros abuelos utilizaban su propia experiencia para anticipar variaciones en el clima. Si se lo analiza desde una perspectiva vinculada con la IA, podría decirse que la experiencia transmitida de generación en generación funcionaba como un ***dataset*** construido de manera oral. Ante una nueva situación —por ejemplo, dolor en las articulaciones—, el «algoritmo», entendido como la experiencia acumulada, permitía predecir un cambio climático. Cuando la predicción se cumplía, se reforzaba como conocimiento válido dentro del entorno familiar, como ocurrió, por ejemplo, en quien escribe este material.

En consecuencia, frente a la pregunta que titula este apartado, puede afirmarse que no existía inteligencia artificial en el sentido moderno, pero sí se realizaban predicciones basadas en información empírica.

En la actualidad, es posible cuantificar la precisión de los algoritmos de ***machine learning*** y validar los modelos mediante métricas específicas. Los ajustes realizados durante el entrenamiento pueden generar sobreajuste o introducir sesgos, por lo que resulta necesario contar con un modelo adecuadamente entrenado que permita minimizar falsos positivos y falsos negativos, ambos cuantificables.

Esta capacidad de medición y validación es precisamente la diferencia con las prácticas de nuestros abuelos. Si sus predicciones no se cumplían, probablemente el hecho no quedaba registrado ni sistematizado, ya que la verificación formal no constituía una prioridad.

CONTINUAR

# 2. Algoritmos de machine learning

**Los algoritmos de machine learning (aprendizaje automático) son modelos computacionales que aprenden patrones y relaciones a partir de datos sin ser programados explícitamente, mejorando su precisión con la experiencia. Estos algoritmos se clasifican según el tipo de aprendizaje que emplean. De manera general, se distinguen tres grandes categorías: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. Cada una responde a una forma diferente de utilizar los datos y de abordar los problemas de predicción o análisis.A continuación, se presentarán los principales tipos de aprendizaje y los algoritmos más utilizados en cada caso.**

1

## Aprendizaje supervisado

El aprendizaje supervisado se basa en datos etiquetados, es decir, conjuntos en los que se conoce previamente el resultado esperado. El modelo aprende a partir de esos ejemplos y luego realiza predicciones sobre nuevos datos.Dentro de esta categoría se encuentran los siguientes algoritmos principales:

- **Regresión lineal y regresión logística**

La regresión lineal y la regresión logística son métodos utilizados para predecir valores numéricos o clasificar datos. Se emplean en ámbitos como negocios, ciencia y economía

para determinar tendencias, pronosticar valores futuros y analizar la relación entre variables. En el caso de la regresión lineal, pueden distinguirse dos tipologías:

- **Regresión lineal simple**. Utiliza una sola variable independiente para predecir la variable dependiente.
- **Regresión lineal múltiple**. Emplea dos o más variables independientes.

El método de mínimos cuadrados es la técnica utilizada para determinar la recta de mejor ajuste, minimizando la suma de los cuadrados de las diferencias entre los valores observados y los valores predichos por el modelo.

- **Árboles de decisión**

Un árbol de decisión es una herramienta gráfica y analítica que modela decisiones y sus posibles resultados mediante una estructura ramificada. Se utiliza tanto en el análisis empresarial como en *machine learning*, para tareas de clasificación y regresión, dividiendo los datos en subconjuntos que permiten generar predicciones. Su estructura se compone de los siguientes elementos:

- **Nodo raíz**. Representa el punto inicial del problema.
- **Nodos de decisión**: indican puntos en los que se elige una acción entre varias alternativas.
- **Nodos de probabilidad**: representan eventos aleatorios con probabilidades asociadas.
- **Ramas**: conectan los nodos y muestran las posibles decisiones o resultados.

- **Nodos terminales o hojas**: contienen el resultado final de cada recorrido.

- *Random forest* **(bosques aleatorios)**

*Random forest* es un algoritmo de aprendizaje supervisado que combina múltiples árboles de decisión con el objetivo de mejorar la precisión y la estabilidad del modelo. Funciona mediante un enfoque de conjunto (*ensemble*), en el que varios árboles generan predicciones de manera independiente y luego se combinan mediante votación (en clasificación) o promedio (en regresión). Este procedimiento reduce la varianza y disminuye el riesgo de sobreajuste. Entre sus principales características se encuentran las siguientes:

- Utiliza múltiples árboles entrenados con diferentes subconjuntos de datos.
- Reduce la varianza al promediar los resultados.
- Presenta buen desempeño frente a datos con ruido o incompletos.
- Puede aplicarse tanto a problemas de clasificación como de regresión.

- **SVM (máquinas de soporte vectorial)**

Las máquinas de soporte vectorial (SVM, por sus siglas en inglés) son algoritmos de aprendizaje supervisado utilizados principalmente para tareas de clasificación y regresión. Su finalidad es separar los datos encontrando el margen más amplio posible entre las distintas clases.En cuanto a su funcionamiento, el objetivo consiste en hallar un hiperplano —una línea en dos dimensiones, un plano en tres dimensiones o una superficie en dimensiones superiores—

que divida los datos en diferentes categorías.Entre sus conceptos fundamentales se encuentran los siguientes:

- **Margen máximo**. No solo busca separar los datos, sino hacerlo dejando el mayor espacio posible entre las clases.
- **Vectores de soporte**: son los puntos de datos más cercanos al hiperplano y determinan la posición y orientación de la frontera de decisión.
- **Truco del *kernel***: cuando los datos no son linealmente separables, se utiliza una función denominada «kernel» para proyectarlos a un espacio de mayor dimensión en el que puedan separarse adecuadamente.

En relación con sus aplicaciones, se emplean ampliamente en reconocimiento de imágenes y rostros, clasificación de texto y detección de *spam*, así como en bioinformática, por ejemplo, para la clasificación de proteínas o la detección de cáncer.Entre sus ventajas se destaca su eficacia en espacios de alta dimensión, su resistencia al sobreajuste (*overfitting*) y su eficiencia en el uso de memoria, ya que trabaja únicamente con los vectores de soporte. Como limitación, puede implicar un costo computacional elevado en conjuntos de datos muy grandes y su interpretación resulta menos directa que la de un árbol de decisión.

- **KNN (*k-nearest neighbors*)**

KNN es un algoritmo de aprendizaje supervisado que clasifica nuevos datos según su proximidad a otros previamente etiquetados. Para ello, identifica los k vecinos más cercanos mediante una medida de distancia y asigna la clase predominante entre

ellos. Es un método sencillo e intuitivo, aunque puede volverse costoso en tiempo de cálculo cuando el conjunto de datos es muy

2 **Aprendizaje no supervisado**

El aprendizaje no supervisado trabaja con datos no etiquetados y tiene como objetivo identificar patrones, agrupaciones o estructuras ocultas dentro de la información disponible. A diferencia del aprendizaje supervisado, no parte de respuestas conocidas, sino que busca descubrir relaciones internas en los datos, lo que lo hace útil para segmentación, detección de anomalías y reducción de dimensionalidad. A continuación, se presentan los principales algoritmos utilizados en este tipo de aprendizaje.

- *K-means clustering* **(agrupación)**

K-means es un algoritmo de aprendizaje no supervisado que divide un conjunto de datos en k grupos o clústeres, buscando maximizar la similitud interna de cada grupo. Para ello, asigna de manera iterativa los puntos de datos al centroide más cercano y recalcula dichos centroides hasta que los clústeres dejan de modificarse de forma significativa.El objetivo consiste en agrupar los objetos en k conjuntos a partir de sus características, minimizando la suma de las distancias —generalmente la distancia euclidiana— entre cada punto y el centroide de su grupo.El procedimiento es iterativo. En primer lugar, se inicializan k centroides, normalmente de manera aleatoria. Luego, cada dato se asigna al centroide más cercano. A continuación, los centroides se recalculan como el promedio de los puntos pertenecientes a cada clúster. Estos pasos se repiten hasta alcanzar la convergencia, es decir, cuando los centroides ya no presentan cambios relevantes.El número de clústeres debe definirse previamente, ya que el valor de k es un parámetro establecido por el usuario antes de ejecutar el algoritmo. Para determinar un

número adecuado de clústeres, suele emplearse el método del codo («elbow method»), que analiza la suma de cuadrados dentro de cada clúster, también denominada variación intraclúster.K-means se utiliza con frecuencia en segmentación de clientes, compresión de imágenes, detección de anomalías y análisis exploratorio de datos. Entre sus limitaciones se encuentra su sensibilidad a valores atípicos (*outliers*), la necesidad de definir k de antemano y la dificultad para manejar clústeres con formas no globulares.

- **PCA (análisis de componentes principales)**

El análisis de componentes principales (PCA, por sus siglas en inglés) es una técnica de aprendizaje no supervisado utilizada para reducir la dimensionalidad de conjuntos de datos amplios, simplificándolos sin perder la información más relevante. Transforma variables correlacionadas en nuevas variables no correlacionadas denominadas componentes principales, ordenadas según la proporción de varianza que explican.Su finalidad es disminuir el número de variables presentes en una tabla de datos, lo que facilita la visualización en dos o tres dimensiones y mejora la eficiencia computacional. La transformación es ortogonal, ya que los componentes principales resultantes son independientes entre sí. El primer componente explica la mayor parte de la varianza total, el segundo la siguiente proporción más alta, y así sucesivamente.Al tratarse de un método no supervisado, no utiliza una variable objetivo, sino que se enfoca en la estructura interna de los datos.Entre sus aplicaciones se encuentran la visualización de datos de alta dimensionalidad en espacios reducidos, la compresión de imágenes mediante la conservación de los componentes más significativos, el filtrado de ruido al eliminar información redundante y el preprocesamiento previo a la aplicación de algoritmos de agrupamiento o clasificación.

- **Redes neuronales (*deep learning*)**

Las redes neuronales (*deep learning*) son modelos de inteligencia artificial inspirados en el cerebro humano. Están compuestas por capas de neuronas interconectadas que aprenden a reconocer patrones complejos en grandes conjuntos de datos para realizar tareas como clasificación, predicción y procesamiento del lenguaje natural (PLN). Funcionan mediante el entrenamiento con datos para ajustar sus pesos o parámetros y minimizar errores, lo que les permite identificar información en imágenes, textos y sonidos. Son la base de asistentes virtuales, diagnóstico médico y reconocimiento facial, entre otras aplicaciones.Desde el punto de vista estructural, presentan los siguientes elementos:

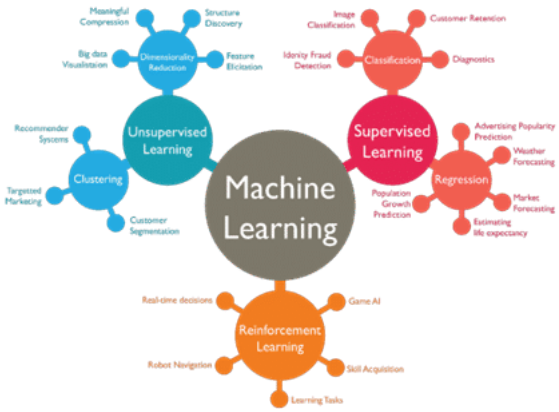
- Estructura en capas: reciben datos en una capa de entrada, los procesan en una o varias capas ocultas y producen una salida en una capa final.
- Interconexiones: las neuronas se comunican a través de conexiones, cada una con un peso que determina la importancia de la información.
- Aprendizaje por retropropagación (*backpropagation*): se alimentan datos conocidos, se compara la salida generada con la esperada y se ajustan los pesos para reducir el error en un proceso iterativo.
- Función de activación: convierte las entradas ponderadas en la activación de una neurona para la siguiente capa.

Entre los tipos principales se encuentran las redes neuronales convolucionales (CNN), utilizadas en reconocimiento de imágenes; las redes neuronales recurrentes (RNN), empleadas en datos secuenciales como texto y voz; y los transformadores (*transformers*), arquitecturas que impulsan modelos de lenguaje de gran

tamaño (LLM).Se aplican en visión por computadora, procesamiento del lenguaje natural, sistemas de recomendación, ciberseguridad y medicina.

La siguiente figura sintetiza la clasificación de los principales enfoques del machine learning, integrando aprendizaje supervisado, no supervisado y por refuerzo dentro de un mismo marco conceptual.

**Figura 2. Tipos de aprendizaje en machine learning y sus principales aplicaciones**




Fuente: ICCSI, s.f., https://goo.su/5iHNCcD

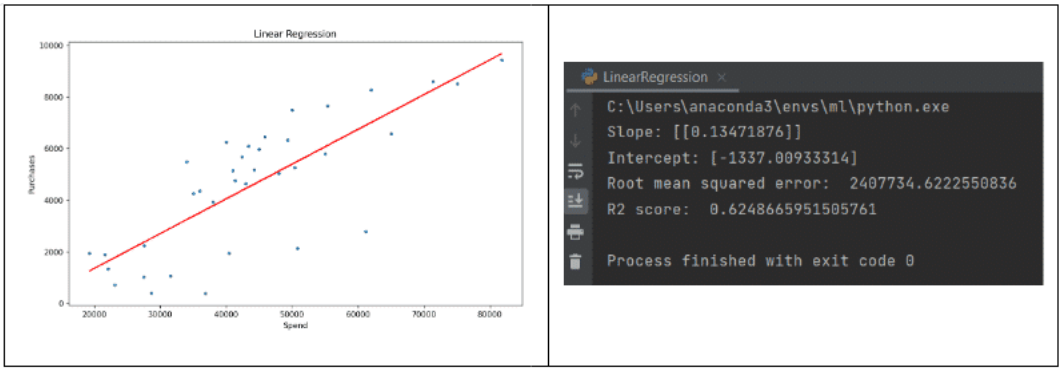
CONTINUAR

# 3. Modelos supervisados

## Regresión

Este modelo supervisado se utiliza cuando se busca predecir un valor numérico. Responde a preguntas del tipo: ¿cuántos productos se venderán el mes próximo?, ¿qué beneficios se obtendrán el próximo año? Este tipo de algoritmo intenta estimar un valor cuantitativo a partir del comportamiento previo de los datos.

**Figura 3. Ejemplo de regresión lineal y métricas del modelo**




Fuente: Ruiz, 2022, https://goo.su/XQl7L2z

En la imagen se presenta, a la izquierda, la representación gráfica de una regresión lineal ajustada a un conjunto de datos y, a la derecha, los valores obtenidos tras la ejecución del modelo.

En la imagen de la derecha, correspondiente a la segunda parte del código desarrollado, se observan varios valores cuyo significado es el siguiente:

**Slope:**

La fórmula en la que se basa el algoritmo de regresión lineal es y = mx + b. En este caso, el *slope* (pendiente de la

recta) corresponde al coeficiente m.

**Intercept:** _

representa el valor constante b. Tanto el *slope* como el *intercept* expresan la relación lineal entre dos variables.

**Root mean squared error (RMSE):** _

indica la desviación estándar de los residuos, es decir, cuánto varían los datos respecto de la línea de regresión. Dado que su valor depende de la escala de los datos, puede evaluarse mediante la expresión (RMSE / promedio de y) × 100 % para determinar si el modelo resulta aceptable. En muchos casos, un valor inferior al 10 % se considera adecuado.

**$R^2$:** _

mide el grado de relación entre el modelo generado y la variable dependiente. No existe un valor único que determine su calidad, aunque valores superiores a 0,6 suelen interpretarse como una correlación alta, dependiendo del ámbito de aplicación.

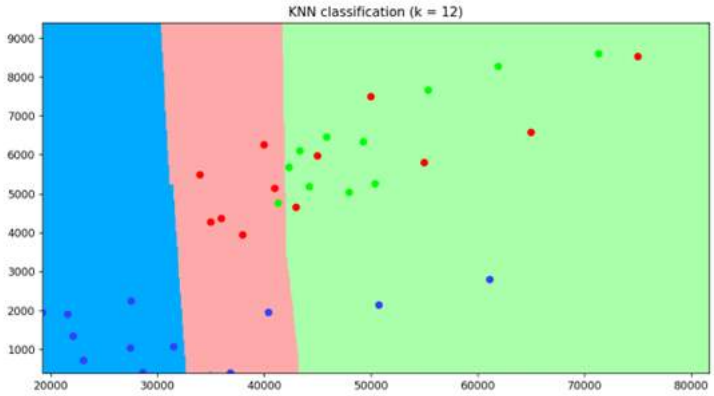
## Clasificación

La clasificación es un algoritmo supervisado orientado a predecir valores categóricos. Permite responder a preguntas como: ¿están los clientes satisfechos?, ¿será devuelto el producto? A diferencia de la regresión, donde el resultado es un valor continuo, en la clasificación los resultados son discretos, lo que posibilita segmentar los datos en categorías definidas.

En el siguiente ejemplo práctico se desarrolla un algoritmo KNN, en el que se ha calculado previamente la mejor aproximación y el valor óptimo de K. Este valor representa la cantidad de vecinos más cercanos que se consideran para asignar una clase a un nuevo dato.

Además, existen otros algoritmos, como los árboles de decisión, que pueden aplicarse tanto a problemas de regresión como de clasificación. Este tipo de modelo suele emplearse también como complemento para contrastar o validar el desempeño de otros enfoques.

**Figura 4. Ejemplo de clasificación mediante KNN (k = 12)**




Fuente: Ruiz, 2022, https://goo.su/XQl7L2z

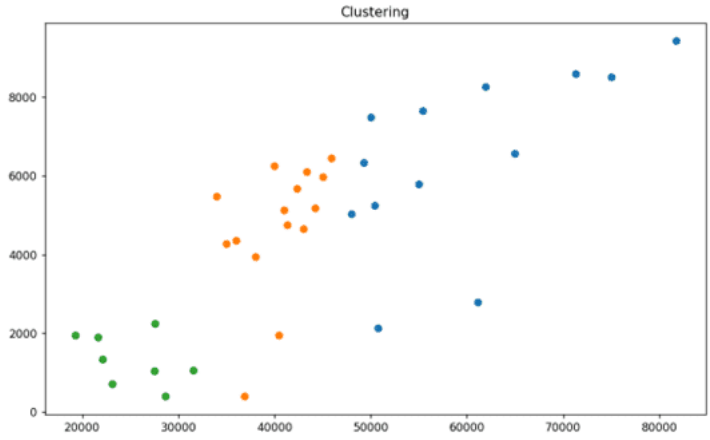
## Modelos no supervisados

### *Clustering*

Aunque puede parecer similar a la clasificación, ya que también organiza los datos en grupos, el *clustering* es un algoritmo no supervisado y no requiere información previamente etiquetada. Su objetivo consiste en identificar agrupaciones naturales dentro de los datos a partir de sus características.

Permite responder a preguntas como: ¿cómo segmentar clientes según su nivel de gasto?, ¿cómo agrupar productos con comportamientos similares? En este enfoque, los grupos se forman a partir de la proximidad o similitud entre los datos, sin contar con categorías definidas de antemano.

**Figura 5. Ejemplo de agrupamiento mediante *clustering***

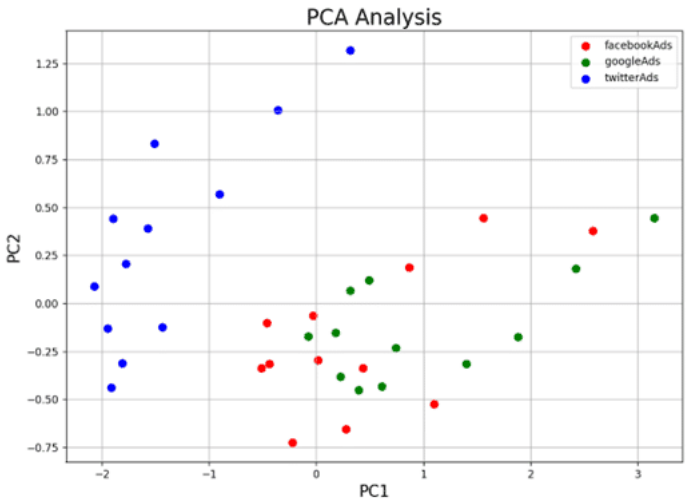



Fuente: Ruiz, 2022, https://goo.su/XQl7L2z

## Reducción de dimensionalidad

La reducción de dimensionalidad se orienta a disminuir el número de variables con el objetivo de organizar los datos a partir de patrones más simples y resaltar sus características esenciales. En este proceso, se transforman múltiples variables originales en un conjunto menor de componentes que conservan la mayor parte de la información relevante. Al tratarse de un modelo no supervisado, no se dispone previamente de etiquetas que indiquen la categoría de los datos, por lo que el análisis se basa exclusivamente en la estructura interna del conjunto de información.

**Figura 6. Ejemplo de reducción de dimensionalidad mediante PCA**




Fuente: Ruiz, 2022, https://goo.su/XQl7L2z

CONTINUAR

# Referencias

**ICCSI**, (s.f.). Big data, machine learning y realidad aumentada: una revolución tecnológica. https://iccsi.com.ar/big-data-machine-learning-inteligencia-artificial-realidad-aumentada/

**Farmalastic**, (s.f.). *¿Por qué me duelen más las articulaciones cuando cambia el tiempo?* https://farmalastic.cinfa.com/blog/por-que-me-duelen-mas-las-articulaciones-cuando-cambia-el-tiempo

**Ruiz, J.** (2022). *Modelos Machine Learning con Python.* https://josebaruiz.com/modelos-machine-learning-con-python/

**Referencias bibliográficas de consulta**

**IBM,** (s.f.). *¿Qué son las redes neuronales?* https://www.ibm.com/es-es/think/topics/neural-networks

**Müller, A. C., & Guido, S.** (2016). *Introduction to machine learning with Python: A guide for data scientists.* O'Reilly Media. https://www.nrigroupindia.com/e-book/Introduction%20to%20Machine%20Learning%20with%20Python%20(%20PDFDrive.com%20)-min.pdf

**W3 School**, (s.f.). *Python Tutorial.* https://www.w3schools.com/python/

**W3 School**, (s.f.). *Machine learning.* https://www.w3schools.com/python/python_ml_getting_started.asp

CONTINUAR