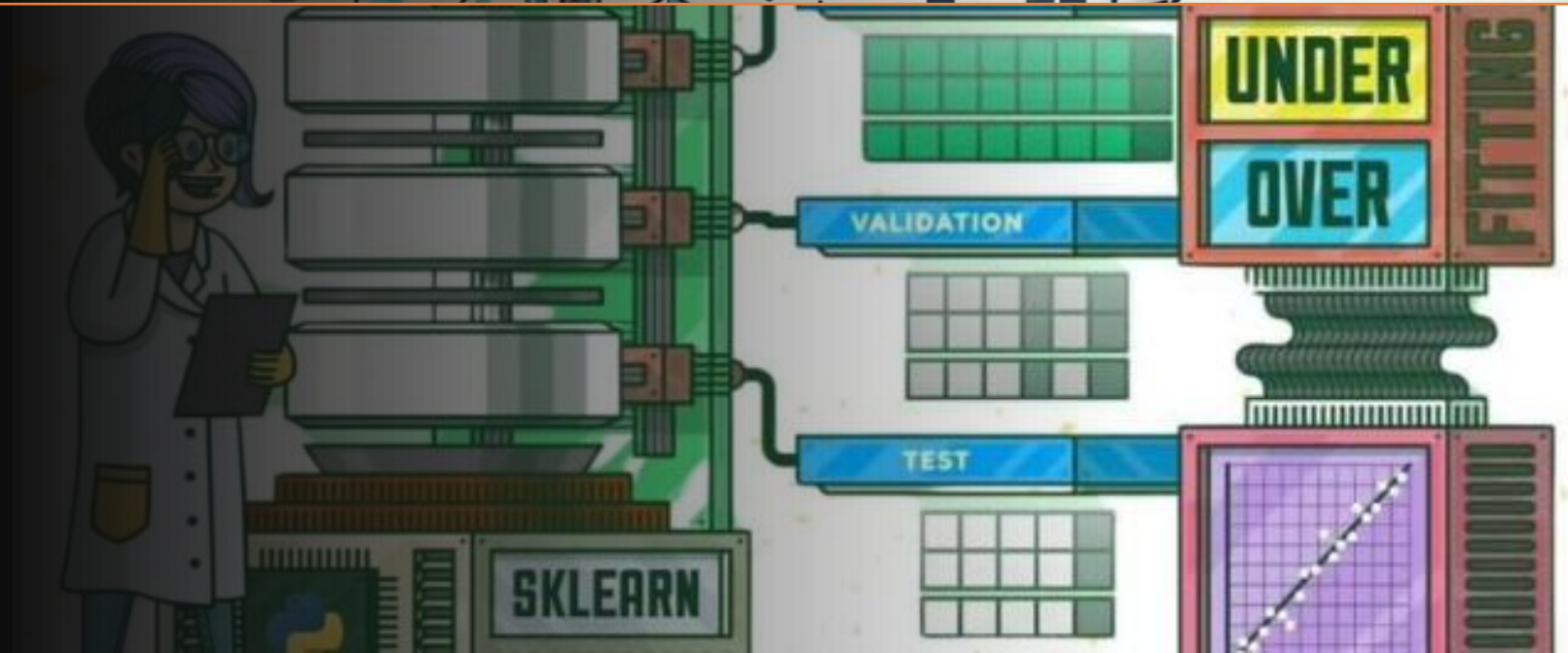


# LIBRERÍAS



Proporcionan una gran cantidad de funciones y métodos preconstruidos que los desarrolladores pueden utilizar sin necesidad de reinventar la rueda.

## Ejemplos de Bibliotecas Populares y sus Usos

- **NumPy**: Para cálculos numéricos y operaciones con matrices.
- **Pandas**: Para manipulación y análisis de datos.
- **Matplotlib**: Para visualización de datos.
- **Scikit-learn**: Para machine learning y análisis predictivo.
- **TensorFlow y Keras**: Para desarrollo de redes neuronales y deep learning.
- **Requests**: Para realizar solicitudes HTTP.
- **BeautifulSoup**: Para web scraping.
- **Django y Flask**: Para desarrollo web.

```
# Con Librerías
```

```
import numpy as np
```

```
# Datos de ejemplo
```

```
valores = np.array([10, 20, 30, 40, 50])
```

```
# Calcular y mostrar resultados
```

```
media = np.mean(valores)
```

```
mediana = np.median(valores)
```

```
desviacion_estandar = np.std(valores)
```

```
print(f"Media: {media}")
```

```
print(f"Mediana: {mediana}")
```

```
print(f"Desviación Estándar: {desviacion_estandar}")
```

```
def calcular_media(valores):  
    return sum(valores) / len(valores)
```

```
def calcular_mediana(valores):  
    valores_ordenados = sorted(valores)  
    n = len(valores)  
    mitad = n // 2  
    if n % 2 == 0:  
        return (valores_ordenados[mitad - 1] + valores_ordenados[mitad]) / 2  
    else:  
        return valores_ordenados[mitad]
```

```
def calcular_desviacion_estandar(valores):  
    media = calcular_media(valores)  
    varianza = sum((x - media) ** 2 for x in valores) / len(valores)  
    return math.sqrt(varianza)
```

```
# Datos de ejemplo
```

```
valores = [10, 20, 30, 40, 50]
```

```
# Calcular y mostrar resultados
```

```
media = calcular_media(valores)
```

```
mediana = calcular_mediana(valores)
```

```
desviacion_estandar = calcular_desviacion_estandar(valores)
```

```
print(f"Media: {media}")
```

```
print(f"Mediana: {mediana}")
```

```
print(f"Desviación Estándar: {desviacion_estandar}")
```

```
C: > Users > esmendoza > AppData > Roaming > jupyter > runtime > import pandas as p

1 import pandas as pd
2
3 # Crear un DataFrame de ejemplo
4 datos = pd.DataFrame({"manzana": [3, 2], "peras": [1, 4]})
5
6 # Mostrar el DataFrame
7 print(datos)
8
```

> ▾ TERMINAL



```
PS C:\Users\esmendoza> & C:/Users/esmendoza/AppData/Local/Programs/Python/Python312/python.exe "c:/User
s/esmendoza/AppData/Roaming/jupyter/runtime/import pandas as pd.py"
```

	manzana	peras
0	3	1
1	2	4

```
PS C:\Users\esmendoza> █
```

## Conceptos Básicos:

### float

```
>>> 'manzana'
'manzana'
>>> type('manzana')
<class 'str'>
```

#### 1. int => Scalar:

- **Descripción:** Un `int` en Python es un número entero. En matemáticas y ciencia de datos, un `scalar` es un solo valor numérico.
- **Ejemplo:**

python


 Copiar código

```
scalar = 5 # Este es un escalar, que es un entero.
```

#### 2. list => arrays:

- **Descripción:** Una `list` en Python es una colección ordenada y mutable de elementos. Un `array` es una estructura de datos que almacena elementos del mismo tipo y es más eficiente.
- **Ejemplo:**

python


 Copiar código

```
my_list = [1, 2, 3, 4, 5] # Esto es una lista en Python.
import numpy as np
my_array = np.array([1, 2, 3, 4, 5]) # Esto es un array en NumPy.
```

### 3. arrays == nDim 1, 2, 3:

- **Descripción:** Los arrays pueden tener una o más dimensiones:
  - 1D (unidimensional): Array lineal.
  - 2D (bidimensional): Matriz.
  - 3D (tridimensional): Cubo de datos.
- **Ejemplo:**

python

 Copiar código

```
# Array 1D
array_1d = np.array([1, 2, 3, 4, 5])

# Array 2D (Matriz)
array_2d = np.array([[1, 2, 3], [4, 5, 6]])

# Array 3D
array_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
```



Array 1D:  
[1 2 3 4 5]

Array 2D (Matriz):  
[[1 2 3]  
[4 5 6]]

Array 3D:  
[[[1 2]  
[3 4]]  
[[5 6]  
[7 8]]]

# Estructuras Más Complejas:

## 1. matrices:

- Descripción: Las matrices son arrays bidimensionales.
- Ejemplo:

python

```
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```



```
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
print(matrix)
```



```
[[1 2 3]
```

- Interpretación de las dimensiones:

- La primera dimensión tiene 2 elementos.
- Cada uno de esos elementos es una matriz de 2x2.
- Por lo tanto, el tensor es un cubo de datos que puede visualizarse como 2 matrices de 2x2.

```
tf.Tensor(  
[[[1 2]  
[3 4]]
```

de

```
[[5 6]  
[7 8]]], shape=(2, 2, 2), dtype=int32)
```

python

```
import tensorflow as tf  
import torch  
  
# Tensor en TensorFlow  
tensor_tf = tf.constant([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])  
  
# Tensor en PyTorch  
tensor_pt = torch.tensor([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
```



## Introducción a pandas

**Pandas** es una biblioteca de Python ampliamente utilizada para el análisis y la manipulación de datos. Ofrece estructuras de datos flexibles y potentes, junto con diversas herramientas para la limpieza, preparación y análisis de datos. Pandas es esencial en la ciencia de datos y el aprendizaje automático, ya que permite a los usuarios trabajar con datos tabulares de manera eficiente.

### Estructuras de Datos en Pandas

Pandas tiene dos estructuras de datos principales: **Series** y **DataFrame**.



# Series

Una **Series** es una matriz unidimensional similar a una columna en una tabla. Cada elemento en una Serie tiene un índice asociado, lo que facilita la recuperación de valores. Las Series pueden contener cualquier tipo de datos (enteros, cadenas, flotantes, etc.).

## Características de Series:

- Unidimensional.
- Etiquetada (indexada).
- Puede contener datos heterogéneos.

Ejemplo de una Serie:

python

 Copiar código

```
import pandas as pd

# Crear una Serie
serie = pd.Series([1, 2, 3, 4, 5])
print(serie)
```

# DataFrame

Un **DataFrame** es una estructura de datos bidimensional, similar a una tabla en una base de datos o una hoja de cálculo de Excel. Está compuesto por múltiples Series alineadas a lo largo de un índice común. Los DataFrames pueden tener múltiples columnas, cada una con un nombre y un tipo de dato específico.

## Características de DataFrame:

- Bidimensional (filas y columnas).
- Etiquetado (indexado) tanto en filas como en columnas.
- Puede contener datos heterogéneos.

```
import pandas as pd

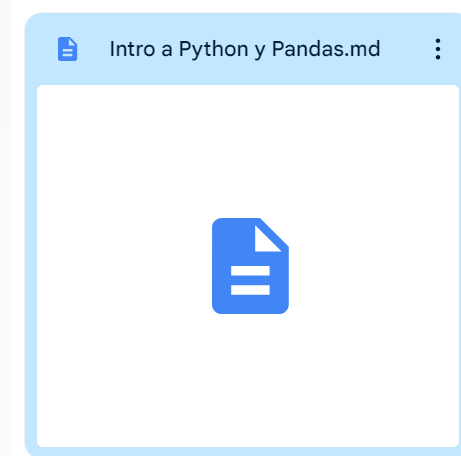
# Crear un DataFrame
data = {
    'Nombre': ['Ana', 'Luis', 'Marta', 'Carlos'],
    'Edad': [23, 45, 34, 25],
    'Ciudad': ['Madrid', 'Barcelona', 'Sevilla', 'Valencia']
}

df = pd.DataFrame(data)
print(df)
```

## Diferencias Clave entre Series y DataFrame

Característica	Series	DataFrame
Dimensión	Unidimensional	Bidimensional
Estructura	Similar a una columna	Similar a una tabla
Indexación	Solo por filas	Por filas y columnas
Homogeneidad	Puede contener datos heterogéneos	Puede contener datos heterogéneos
Ejemplo de uso	<code>pd.Series([1, 2, 3])</code>	<code>pd.DataFrame({'col1': [1, 2]})</code>

# EJEMPLOS PRACTICOS CON PANDAS y NUMPY CLASE



## Básico de la guía

En el siguiente libro encontrarás una implementación práctica de una regresión lineal y logística con dos ejemplos diferentes.

Bagnato, J. (2017). *Aprende Machine Learning en Español Teoría + Práctica*. <https://leanpub.com/aprendeml>[Links to an external site.](#) Capítulo 4 y 5, páginas 25-46.

En el apartado del siguiente libro hay una introducción a *Machine Learning* y el uso de Python. Aprenderás algunas herramientas elementales del análisis de datos y estudiarás algunos de los paquetes más utilizados para analizar datos con Python.

Guido, S., & Müller, A. (2017). *Introduction to Machine Learning with Python*. Sebastopol, CA, USA: O'Reilly Media, Inc. Capítulo 1 y 2, páginas 1-29.

En el material de Zainea encontrarás un ejercicio práctico para estudiar diferentes métodos y utilidades del paquete pandas de Python.

Zainea Maya, C. I. (2022). *Seminario-de-programación* (Version 1.0.0) [Computer software]. <https://github.com/lzainea/seminario-de-programacion>

Videos de introducción a *machine learning*:

DotCSV. (1 noviembre de 2017). *¿Qué es el Machine Learning? ¿Y Deep Learning?* Un mapa conceptual [Archivo de video]. Youtube. <https://bit.ly/3zbA0sf>[Links to an external site.](#)

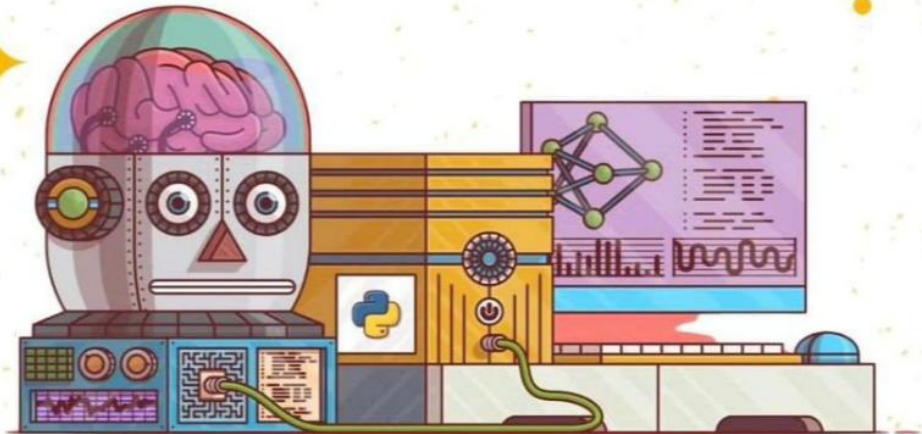
DotCSV. (16 octubre de 2017). *¿Qué es el Aprendizaje Supervisado y No Supervisado?* [Archivo de video]. Youtube. <https://www.youtube.com/watch?v=oT3arRRB2Cw>

## Introducción a Scikit-Learn

**Scikit-Learn** (también conocido como ``sklearn``) es una biblioteca de Python muy utilizada para el aprendizaje automático y el análisis de datos. Proporciona una gama amplia de herramientas eficientes y fáciles de usar para la minería de datos y el análisis de datos. Es particularmente conocida por su simplicidad y consistencia en la interfaz, lo que la hace accesible tanto para principiantes como para expertos.

So, you know how in **Python**, you can do all sorts of cool stuff, like math, data analysis, and even make predictions with machine learning?

Well, **scikit-learn** is like your best buddy for the **machine learning** part.



## Características Clave de Scikit-Learn

### 1. Modelos de Clasificación:

- Ejemplos: K-Nearest Neighbors, SVM, Random Forest, Logistic Regre
- Uso: Asignar etiquetas a instancias.

### 2. Modelos de Regresión:

- Ejemplos: Linear Regression, Ridge Regression, Lasso Regression.
- Uso: Predicción de valores continuos.

### 3. Agrupamiento (Clustering):

- Ejemplos: K-Means, Agglomerative Clustering, DBSCAN.
- Uso: Agrupación de datos no etiquetados en grupos.


### 4. Reducción de Dimensionalidad:

- Ejemplos: PCA (Principal Component Analysis), t-SNE.
- Uso: Reducción de la cantidad de variables de los datos.

## Instalación

Antes de usar Scikit-Learn, asegúrate de que está instalado. Puedes instalarlo usando pip:

```
bash
```

 Copiar código

```
pip install scikit-learn
```




## ``train_test_split``

- **Propósito:** La función ``train_test_split`` se utiliza para dividir un conjunto de datos en dos subconjuntos: uno para entrenamiento y otro para prueba. Esto es fundamental para evaluar el rendimiento de los modelos de machine learning de manera objetiva.

## Cómo se utiliza

### 1. Importación:

python


 Copiar código

```
from sklearn.model_selection import train_test_split
```

## 2. Aplicación:

Supongamos que tienes un dataset con características (features) y etiquetas (labels):

python

 Copiar código

```
X = ... # Conjunto de características (por ejemplo, un DataFrame o una matriz)
y = ... # Conjunto de etiquetas (por ejemplo, un vector o una serie)
```

## 3. División del dataset:

Puedes dividir el dataset en conjuntos de entrenamiento y prueba utilizando

`train_test_split` de la siguiente manera:

python

 Copiar código

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- `X_train`: Conjunto de características para el entrenamiento.
- `X_test`: Conjunto de características para la prueba.
- `y_train`: Conjunto de etiquetas para el entrenamiento.
- `y_test`: Conjunto de etiquetas para la prueba.

python

 Copiar código

```
from sklearn.model_selection import train_test_split

# Supongamos que tienes los siguientes datos:
X = [[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]]
y = [0, 1, 0, 1, 0]

# Dividir el dataset en entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Resultados
print("X_train:", X_train)
print("X_test:", X_test)
print("y_train:", y_train)
print("y_test:", y_test)
```

Esto generará una división del dataset en subconjuntos de entrenamiento y prueba, lo cual es esencial para evaluar el rendimiento de los modelos de machine learning y evitar el sobreajuste.

```
# importing modules and packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, explained_
from sklearn import preprocessing
```

# Ejemplos prácticos de tablas : [Explorar en casa](#)

## Carga de datos

Para cargar los datos debe hacer clic en el panel izquierdo en el botón de exploración de archivos. En ese menu tendra la opción de subir algunos datos.

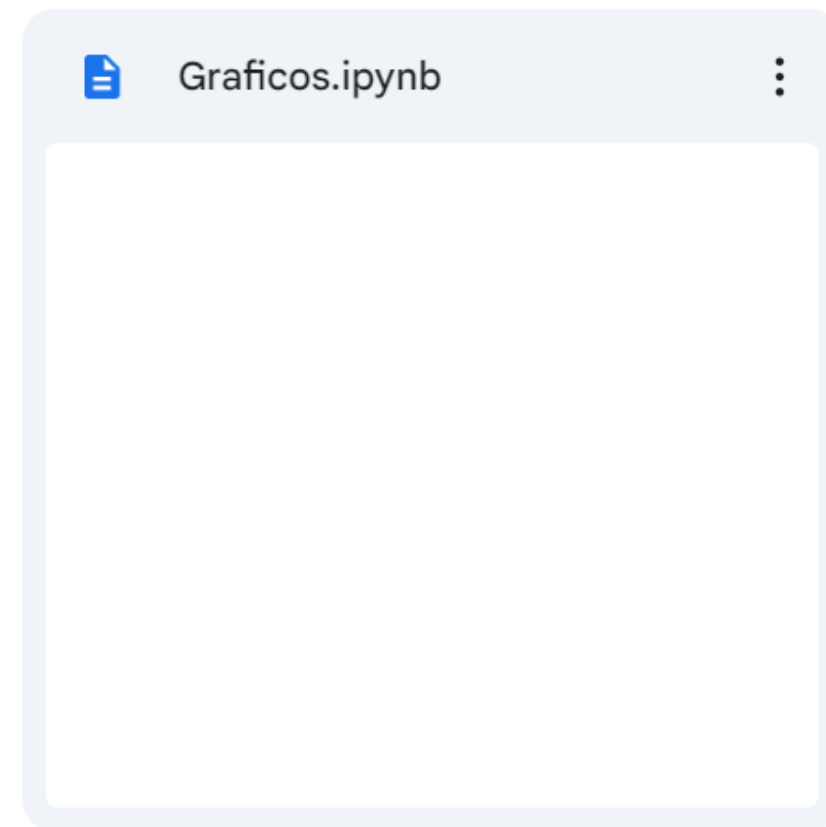
```
import pandas as pd

# Load the dataset from the user's Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Show DataFrame
```

Mounted at /content/drive

```
[3] filename='ifood_df.csv'
path=f'/content/drive/MyDrive/Colab Notebooks/seminario-de-programacion-main/seminario-de-programacion-main/Datos/{filename}'
df = pd.read_csv(path)
```





- ¿Cuál es la diferencia de aprendizaje supervisado y no supervisado?
- ¿Qué paquete permite limpiar y unificar bases de datos con Python? ¿Podría nombrar funciones y métodos útiles para la carga, la limpieza y el procesamiento de datos en Python?
- ¿En qué tipo problemas debo utilizar una regresión lineal?
- ¿En qué tipo problemas debo utilizar una regresión logística?
- ¿En qué consiste y para qué sirve el error cuadrático medio? ¿Es útil para la regresión logística? ¿En la regresión logística, en que se diferencia con la exactitud, la precisión o el puntaje F1?

• Por qué los problemas que se resuelven