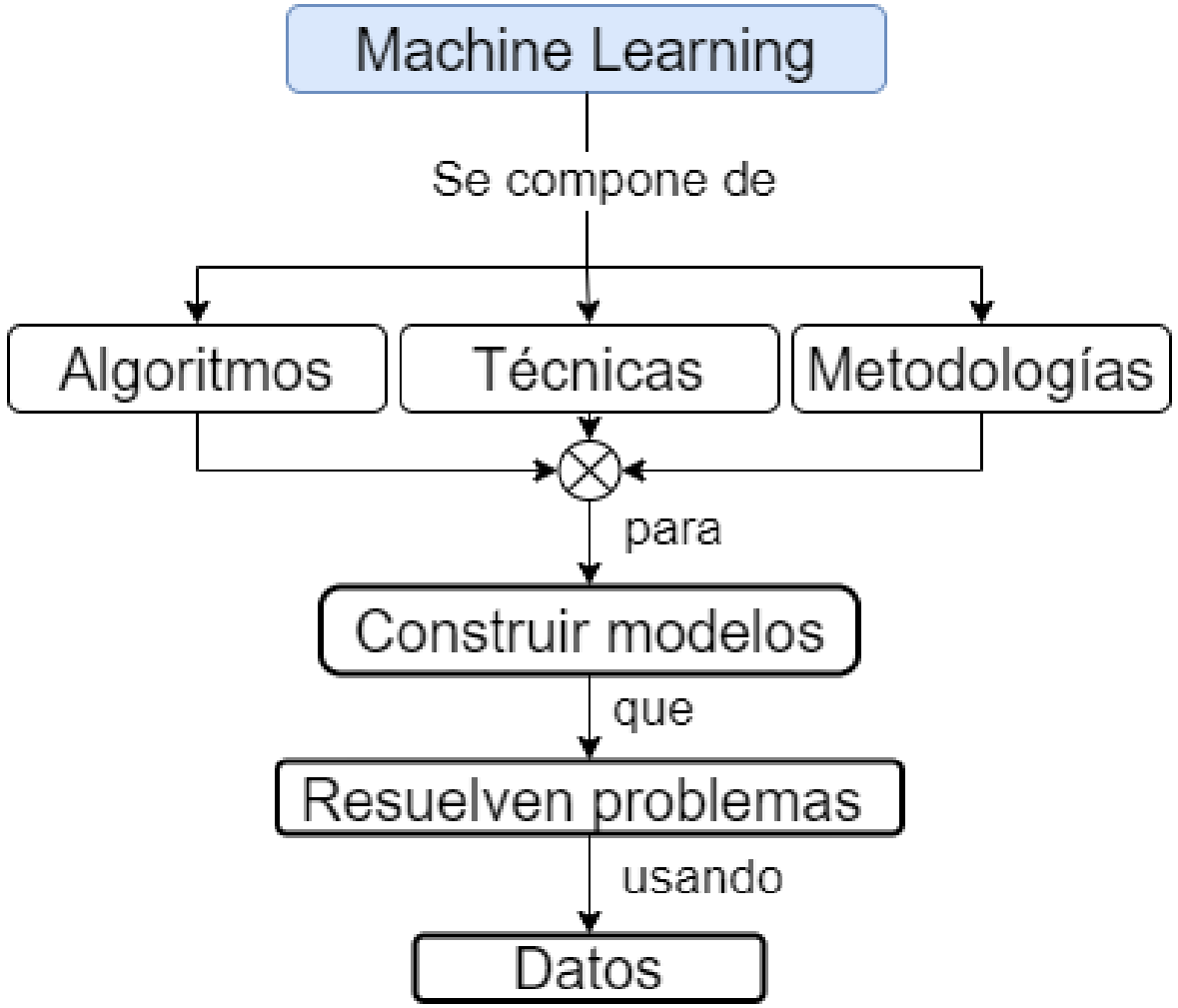
UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

MACHINE LEARNING CON PYTHON

INTRODUCCIÓN

Métodos de Machine Learning



Tipos de problemas

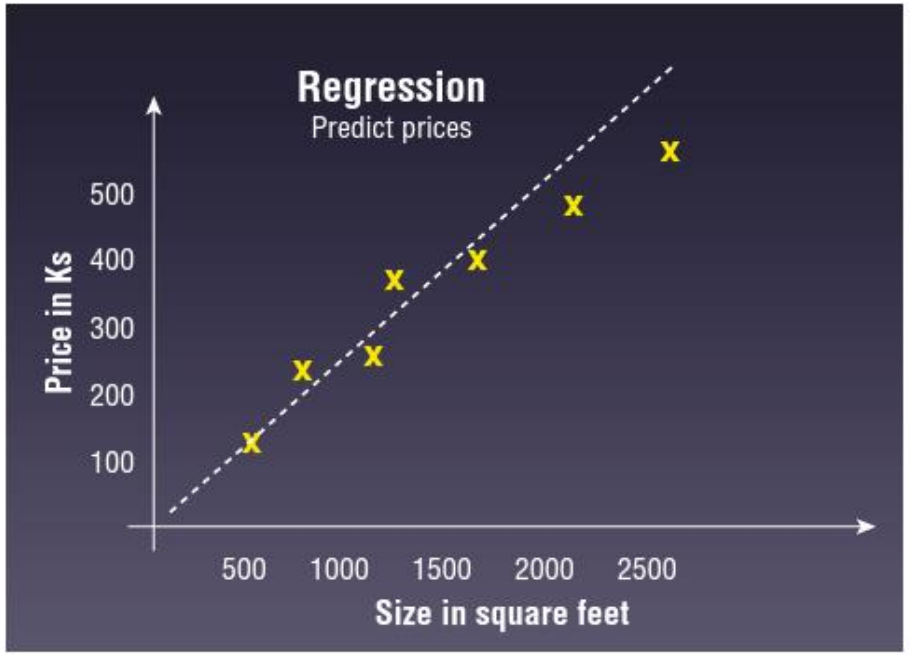


Figure 1.3: Using regression to predict the expected selling price of a house

REGRESIÓN: En estos problemas se ayuda a prever el futuro estimando relaciones entre las variables.

Ejemplos:

- Predecir número de ventas de un ítem particular.
- Predecir la vida útil de un producto.
- Predecir la temperatura de la próxima semana.

Imagen tomada de: LEE, W-M. (2019) Python Machine Learning

Tipos de problemas

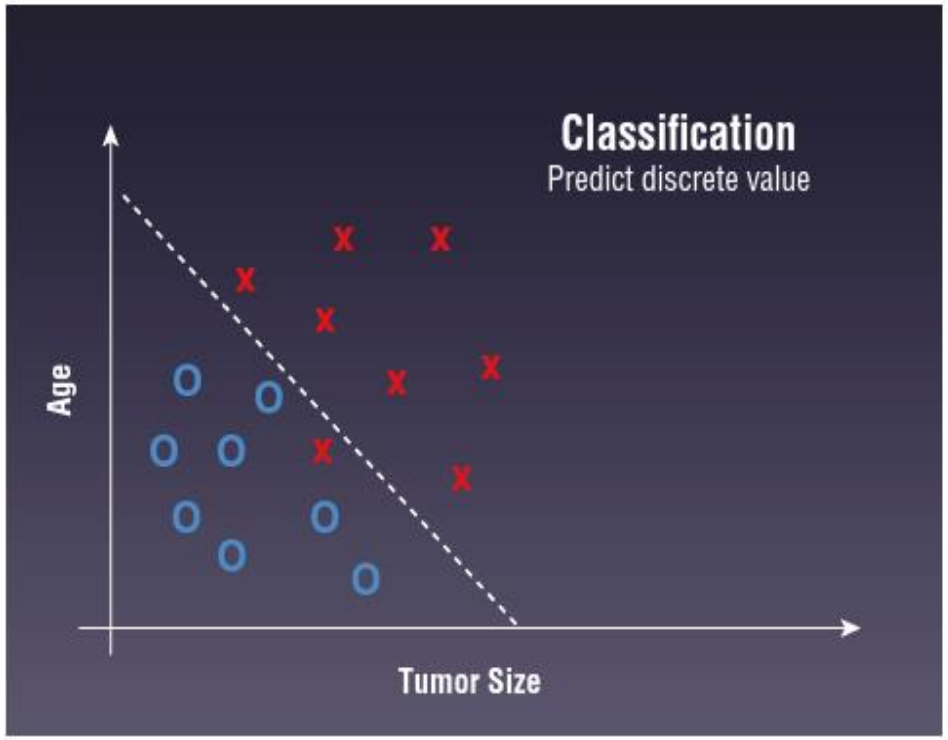


Figure 1.4: Using classification to categorize data into distinct classes

CLASIFICACIÓN: En estos problemas se separan los datos, según alguna característica específica. También sirve para predecir.

Ejemplos:

- Predecir el ganador de las elecciones.
- Clasificar tipos de flores.
- Predecir si un tumor es cancerígeno.

Imagen tomada de: LEE, W-M. (2019) Python Machine Learning

Tipos de problemas

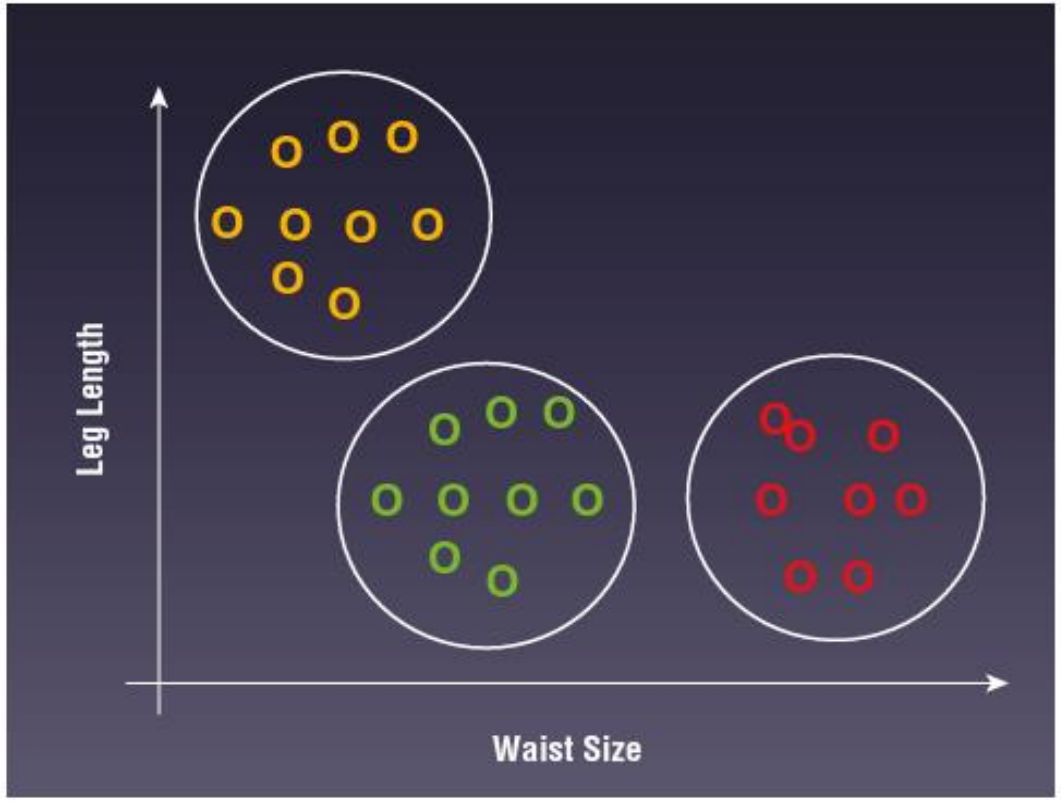


Figure 1.6: Clustering the points into distinct groups

CLUSTERING O AGRUPAMIENTO: En estos problemas se agrupan datos similares dentro de conjuntos

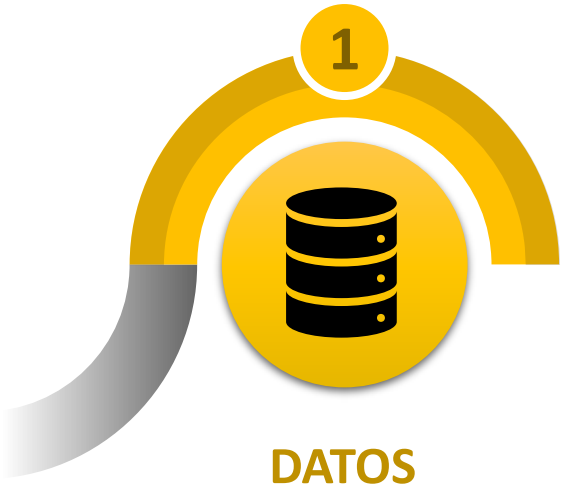
Ejemplos:

- ¿Qué modelos de disco duro fallan de la misma forma?.
- ¿A cuántos clientes les gusta un producto en particular?.

Imagen tomada de: LEE, W-M. (2019) Python Machine Learning

Métodos de Machine Learning

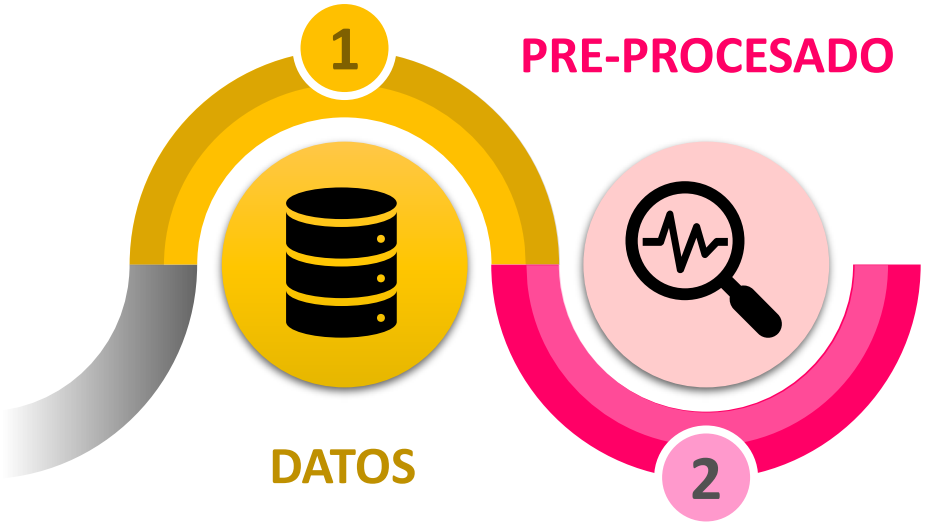
RESUMEN DE PROCESOS



DATOS

Métodos de Machine Learning

RESUMEN DE PROCESOS

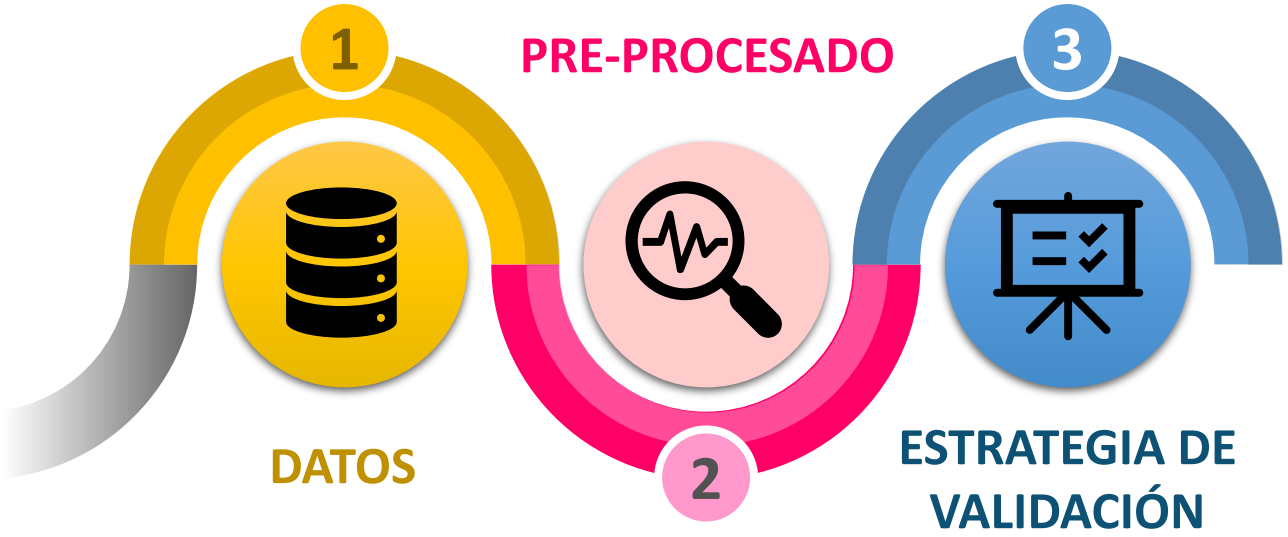


Preprocesado:
Consiste en preparar los datos para que tengan un formato válido. Dentro de las transformaciones más comunes están:

- Conversión de datos tipo texto a numérico.
- Eliminar duplicados.
- Rellenar valores que faltan.

Métodos de Machine Learning

RESUMEN DE PROCESOS



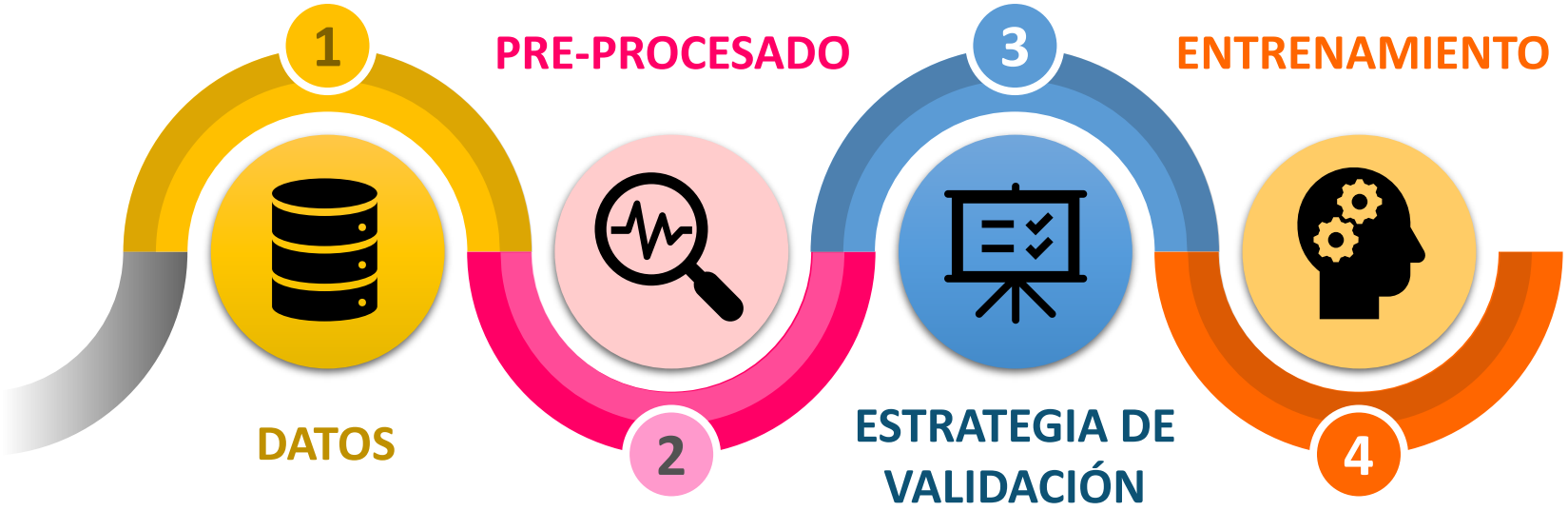
Estrategias de validación:
Esta tarea se realiza con el fin de determinar si el algoritmo funciona eficientemente o no. Para ello se escoge una métrica, que dependerá si el problema es de regresión o de clasificación.

Una estrategia común y simple es dividir los datos en subgrupos:

- Entrenamiento.
- Validación
- Pruebas

Métodos de Machine Learning

RESUMEN DE PROCESOS



Con el conjunto de datos de entrenamiento se enseña al algoritmo cómo debería funcionar. Algunos algoritmos comúnmente usados son: Regresión lineal, regresión logística, KNN (K-Nearest Neighbors), árboles de decisión, entre otros.
Cuando ya se ha entrenado el algoritmo, se usan los datos de validación (no los ha visto el algoritmo) para revisar si funciona bien o no.

Métodos de Machine Learning

RESUMEN DE PROCESOS

Métodos de Machine Learning

Según la cantidad de supervisión humana que tienen los procesos de aprendizaje, los métodos de Machine Learning pueden clasificarse en:

- Aprendizaje supervisado
- Aprendizaje no supervisado

Métodos de Machine Learning

APRENDIZAJE SUPERVISADO

Necesita tener los datos tanto de la variable predictora X como de la variable a predecir Y.

Se entrenan con datos etiquetados (datos de respuesta deseada). Por ejemplo: para detectar si una TC es fraudulenta se hace el entrenamiento con TC tanto válidas como fraudulentas.

Aplicable a problemas de regresión y clasificación.

Métodos de Machine Learning

APRENDIZAJE NO SUPERVISADO

No necesita los datos de la variable objetivo Y.

Usan datos sin etiquetas y busca encontrar relaciones entre ellos.

Aplicable a problemas de Agrupamiento o clustering.

PYTHON

Uso de Python en ML: NumPy

Librería que proporciona estructura de los datos en forma de matriz. Su nombre proviene de Numerical Python.

NumPy array: objeto de matriz N-dimensional.

```
#Ejemplo Numpy
import numpy as np
a=np.array([1,2,3])
print(a)
print()
b=np.array([(1,2,3),(4,5,6)])
print(b)
```

¿Qué creamos en cada caso a y b?

Uso de Python en ML: NumPy

Ventajas de usar NumPy:

- Usan menos memoria que las listas de Python.
- Es más rápido en ejecución.

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz de unos (3 filas y 5 columnas)
matriz1=np.ones((3,5))
print(matriz1)
```

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

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz de ceros (3 filas y 5 columnas)
matriz0=np.zeros((3,5))
print(matriz0)
```

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

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz con un solo valor en todas las posiciones (3 filas y 5 columnas)
matriz_igual = np.full((3,5),5)
print(matriz_igual)
```

```
[[5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]]
```

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz con números aleatorios (3 filas y 5 columnas)
matriz_r=np.random.random((3,5))
print(matriz_r)
```

```
[[0.95093692 0.66173327 0.29796516 0.86076343 0.16734396]
 [0.05927392 0.84817617 0.98610158 0.33722321 0.17245603]
 [0.40025031 0.69878564 0.71827078 0.43490693 0.74810567]]
```

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz con valores que se separan uniformemente
llenarM=np.arange(0,50,10)
print(llenarM)
```

¿Cuáles datos se generan?

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz con valores que se separan uniformemente
llenarM=np.arange(0,50,10)
print(llenarM)
```

¿Cuáles datos se generan?

```
[ 0 10 20 30 40]
```

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz con valores que se separan uniformemente
llenarM=np.arange(0,50,10)
print(llenarM)
```

¿Cuáles datos se generan?

```
[ 0 10 20 30 40]
```

```
llenarM2=np.linspace(0,50,10)
print(llenarM2)
```

Uso de Python en ML: NumPy

Llenar matrices

```
#Crear matriz con valores que se separan uniformemente
llenarM=np.arange(0,50,10)
print(llenarM)
```

¿Cuáles datos se generan?

```
[ 0 10 20 30 40]
```

```
llenarM2=np.linspace(0,50,10)
print(llenarM2)
```

```
[ 0.          5.55555556 11.11111111 16.66666667 22.22222222 27.77777778
 33.33333333 38.88888889 44.44444444 50.        ]
```

Uso de Python en ML: NumPy

Llenar matrices

```
matriz_d=np.eye(4,4)
print(matriz_d)
```

¿Qué tipo de matriz genera?

Uso de Python en ML: NumPy

Llenar matrices

```
matriz_d=np.eye(4,4)
print(matriz_d)
```

¿Qué tipo de matriz genera?

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

Uso de Python en ML: NumPy

Llenar matrices

```
#Parámetros de las matrices
b=np.array([[1,2,3],[4,5,6]])
print(b.ndim)
print(b.dtype)
print(b.size)
print(b.shape)
```

¿Qué se obtiene en cada impresión?

Uso de Python en ML: NumPy

Matrices

```
#Parámetros de las matrices
b=np.array([[1,2,3],[4,5,6]])
print(b.ndim)
print(b.dtype)
print(b.size)
print(b.shape)
```

```
2
int64
6
(2, 3)
```

¿Qué se obtiene en cada impresión?

Uso de Python en ML: NumPy

Matrices

```
b=np.array([[1,2,3],[4,5,6]])
print(b)
print()
a=b.reshape(3,2)
print(a)
```

¿Qué se observa?

Uso de Python en ML: NumPy

Matrices

```
b=np.array([[1,2,3],[4,5,6]])
print(b)
print()
a=b.reshape(3,2)
print(a)
```

¿Qué se observa?

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

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

Uso de Python en ML: NumPy

Matrices

Dada la siguiente matriz, ¿cómo seleccionar e imprimir el elemento que es igual a 9?

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
```

Uso de Python en ML: NumPy

Matrices

Dada la siguiente matriz, ¿cómo seleccionar e imprimir el elemento que es igual a 9?

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
```

```
print(c[1,2])
```

Uso de Python en ML: NumPy

Matrices

Dada la siguiente matriz, ¿cómo imprimir todos los elementos de la primera fila?

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
```

Uso de Python en ML: NumPy

Matrices

Dada la siguiente matriz, ¿cómo imprimir todos los elementos de la primera fila?

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
```

```
#imprimir toda la fila 1
print(c[0,:])
```

Uso de Python en ML: NumPy

Matrices

¿Qué producen las siguientes funciones?

```
c=np.array(((1,2,3,4,5,6),(7,8,9,10,11,12)))
print(c)
print(c.min())
print(c.max())
print(c.sum())
print(c.sum(0))
print(c.sum(1))
```

Uso de Python en ML: NumPy

Matrices

¿Qué producen las siguientes funciones?

```
c=np.array(((1,2,3,4,5,6),(7,8,9,10,11,12)))
print(c)
print(c.min())
print(c.max())
print(c.sum())
print(c.sum(0))
print(c.sum(1))
```

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
1
12
78
[ 8 10 12 14 16 18]
[21 57]
```

Uso de Python en ML: NumPy

Matrices

Obtener la desviación estándar de los datos almacenados en una matriz

```
c=np.array(((1,2,3,4,5,6),(7,8,9,10,11,12)))
print(np.std(c))
```

```
3.452052529534663
```

Uso de Python en ML: NumPy

Matrices

```
#Operaciones básicas de matrices
A=np.array(((1,2),(3,4)))
B=np.array(((9,8),(7,6)))

print(A, "\n")
print(B, "\n")
print(A+B, "\n")
print(A-B, "\n")
print(A*B, "\n")
print(A.dot(B), "\n")
```

Uso de Python en ML: NumPy

Matrices

```
#Operaciones básicas de matrices
A=np.array(((1,2),(3,4)))
B=np.array(((9,8),(7,6)))

print(A, "\n")
print(B, "\n")
print(A+B, "\n")
print(A-B, "\n")
print(A*B, "\n")
print(A.dot(B), "\n")
```

```
[[1 2]
 [3 4]]

[[9 8]
 [7 6]]

[[10 10]
 [10 10]]

[[-8 -6]
 [-4 -2]]

[[ 9 16]
 [21 24]]

[[23 20]
 [55 48]]
```

Uso de Python en ML: PANDAS

Es un popular paquete de Python usado en la ciencia de los datos. Permite manipulación, análisis y organización de datos a través de estructuras como el DATAFRAME.

El nombre de la librería viene de Panel Data

Uso de Python en ML: PANDAS

Gracias a esta librería es posible la implementación de 5 procesos básicos necesarios en la ciencia de los datos:

- Cargar
- Preparar
- Modelar
- Manipular
- Analizar

Uso de Python en ML: PANDAS

DATAFRAME:

Es una estructura bidimensional que permite organizar los datos en forma de tabla, donde se destacan como parámetros:

- Las filas
- Las columnas
- Los datos

Uso de Python en ML: PANDAS

La diferencia fundamental entre NumPy y Panda es que en Panda es posible etiquetar las filas y columnas para acceder a los datos de forma más fácil y directa.

Dicho de otra forma, Pandas es como una versión estructurada de NumPy

Uso de Python en ML: PANDAS

```
#Uso de pandas
#Primero: crear el dataframe

import numpy as np
import pandas as pd
data=np.array([["", "Nombre", "Edad"],["Estudiante1", "Anna", 12],["Estudiante2", "María", 9] ])
df= pd.DataFrame(data)
print(df)
print()
print(pd.DataFrame(data=data[1:,1:], index=data[1:,0], columns=data[0,1:]))
```

Uso de Python en ML: PANDAS

```
#Uso de pandas
#Primero: crear el dataframe

import numpy as np
import pandas as pd
data=np.array([["", "Nombre", "Edad"],["Estudiante1", "Anna", 12],["Estudiante2", "María", 9] ])
df= pd.DataFrame(data)
print(df)
print()
print(pd.DataFrame(data=data[1:,1:], index=data[1:,0], columns=data[0,1:]))
```

```
              0       1     2
0                  Nombre  Edad
1  Estudiante1    Anna    12
2  Estudiante2   María     9

             Nombre Edad
Estudiante1   Anna   12
Estudiante2  María    9
```

¡Siempre hacia lo alto!

Uso de Python en ML: PANDAS

```
print("Forma: ", df.shape)
print("Estadísticas: \n", df.describe())
```

```
Forma:  (3, 3)
Estadísticas:
                    0      1     2
count               3      3     3
unique              3      3     3
top     Estudiante2  María  Edad
freq                1      1     1
```

Uso de Python en ML: PANDAS

```
data2=np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9] ])
df2= pd.DataFrame(data2)
print("Estadísticas: \n", df2.describe())
print()
print("Media: \n", df2.mean())
```

Uso de Python en ML: PANDAS

```
data2=np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9] ])
df2= pd.DataFrame(data2)
print("Estadísticas: \n", df2.describe())
print()
print("Media: \n", df2.mean())
```

```
Estadísticas:
              0    1    2
count  3.0  3.0  3.0
mean   4.0  5.0  6.0
std    3.0  3.0  3.0
min    1.0  2.0  3.0
25%    2.5  3.5  4.5
50%    4.0  5.0  6.0
75%    5.5  6.5  7.5
max    7.0  8.0  9.0

Media:
 0    4.0
1    5.0
2    6.0
dtype: float64
```

Uso de Python en ML: PANDAS

```
print(df2.count())
```

```
0    3
1    3
2    3
dtype: int64
```

```
print(df2.max())
```

```
0    7
1    8
2    9
dtype: int64
```

REFERENCIAS BIBLIOGRÁFICAS

Bishop, C. (2006). Pattern recognition and Machine Learning

http://biblio3.url.edu.gt/Libros/2012/esta-AE/21.pdf

Muller, A. and Guido, S. (2017). Introduction to Machine Learning with Python