

Introducción a la Ciencia de Datos con Python



Unidad I

REPASO PYTHON



¿Qué es una clase?

Una clase es un **modelo o prototipo** que define los **atributos y métodos comunes** a todos los objetos de cierto grupo, un **blueprint**.

- **Atributo:** características que tendrán los objetos pertenecientes a la clase.
- **Métodos:** comportamientos que tendrán los objetos pertenecientes a la clase.

¿Qué es un objeto?

Un objeto es una entidad que agrupa un estado y una funcionalidad relacionadas. El estado del objeto se define a través de los atributos, mientras que la funcionalidad se modela a través de los métodos.

```
class Persona:
    def __init__(self, nombre, anio_nac, dni):
        self.nombre = nombre
        self.anio = anio_nac
        self.dni = dni

    def respira(self):
        print(f"{self.nombre} es un ser humano, por lo tanto respira")
```

```
ana = Persona("Ana García", 1990, 34556790)
# para acceder a los atributos y métodos utilizamos la nomenclatura del punto
print(ana.nombre)
ana.respira()
```

- El constructor `__init__` : se ejecuta justo después de crear un nuevo objeto a partir de la clase, proceso que se conoce con el nombre de instanciación.
- Parámetro **self**: el primer parámetro de `__init__` y del resto de métodos de la clase es siempre `self`, y sirve para referirse al objeto actual. Este mecanismo es necesario para poder acceder a los atributos y métodos del objeto diferenciando, por ejemplo, una variable local `mi_var` de un atributo del objeto `self.mi_var`.
- Para crear un objeto se escribe el nombre de la clase seguido de **cualquier parámetro que sea necesario entre paréntesis, excepto self**. Estos parámetros son los que se pasarán al método `__init__`.

Herencia

En un lenguaje orientado a objetos, cuando hacemos que una clase (subclase) herede de otra clase (súper-clase) estamos haciendo que la subclase contenga todos los atributos y métodos que tenía la súper-clase. Esta acción también se suele llamar a menudo “extender una clase”.

Basta con escribir un nuevo método `__init__` para ella que se ejecutaría en lugar del `__init__` de la súperclase. Esto es lo que se conoce como sobrescribir métodos. Ahora bien, puede ocurrir en algunos casos que necesitemos sobrescribir un método de la clase padre, pero que en ese método queramos ejecutar el método de la clase padre porque nuestro nuevo método no necesite más que ejecutar un par de nuevas instrucciones extra. En ese caso usaríamos la sintaxis `super().__init__(*args)` para llamar al método de igual nombre de la clase padre.

```
class Persona:
    def __init__(self, nombre, anio_nac, dni):
        self.nombre = nombre
        self.anio = anio_nac
        self.dni = dni

    def respira(self):
        print(f"{self.nombre} es un ser humano, por lo tanto respira")

class Empleado(Persona):
    def __init__(self, nombre, anio_nac, dni, profesion, cargo):
        super().__init__(nombre, anio_nac, dni)
        self.profesion = profesion
        self.cargo = cargo

    def puesto(self):
        print(f"{self.nombre} es {self.profesion} y tiene el puesto de {self.cargo}")

juan = Empleado("Juan Perez", 1990, 36382114, "contador", "gerente")
```

Herencia múltiple

Si quisiéramos crear una clase y que herede métodos y atributos de más de una clase, la sintaxis es sencilla:

class Nombre(Súperclase1, Súperclase2,...):

Siendo el orden de importancia de izquierda a derecha, es decir que, si hay dos súper-clases con atributos o métodos iguales, se toma el de la que está indicada primera en los parámetros. Esto se llama **herencia múltiple**.

Encapsulación

La encapsulación se refiere a **impedir el acceso a determinados métodos y atributos de los objetos estableciendo así qué puede utilizarse desde fuera de la clase.**

En Python no existen los modificadores de acceso, y lo que se suele hacer es que el acceso a una variable o función viene determinado por su nombre: si el nombre comienza con dos guiones bajos (y no termina también con dos guiones bajos) se trata de una variable o función privada, en caso contrario es pública.

```

class Persona:
    def __init__(self, nombre, anio_nac, dni):
        self.nombre = nombre
        self.anio = anio_nac
        self.__dni = dni

    def respira(self):
        print(f"{self.nombre} es un ser humano, por lo tanto respira")

    def descripcion(self):
        print(f"{self.nombre} nació en el año {self.anio} y su dni es {self.__dni}")

class Empleado(Persona):
    def __init__(self, nombre, anio_nac, dni, profesion, cargo):
        super().__init__(nombre, anio_nac, dni)
        self.profesion = profesion
        self.cargo = cargo

    def puesto(self):
        print(f"{self.nombre} es {self.profesion} y tiene el puesto de {self.cargo}")

juan = Empleado("Juan Perez", 1990, 36382114, "contador", "gerente")
juan.dni = 31000000
juan.descripcion()

```

Juan Perez nació en el año 1990 y su dni es 36382114

Polimorfismo

El polimorfismo se refiere a la **habilidad de objetos de distintas clases de responder al mismo mensaje**. Esto se puede conseguir a través de la herencia: un objeto de una clase derivada es al mismo tiempo un objeto de la clase padre, de forma que allí donde se requiere un objeto de la clase padre también se puede utilizar uno de la clase hija.

```
class Perro:
    def __init__(self, nombre, raza):
        self.nombre = nombre
        self.raza = raza

    def desplazarse(self):
        print(f"{self.nombre} se mueve en 4 patas")

class Pez:
    def __init__(self, nombre, tipo):
        self.nombre = nombre
        self.tipo = tipo

    def desplazarse(self):
        print(f"{self.nombre} se mueve nadando")

canela = Perro("Canela", "de la calle")
nemo = Pez("Nemo", "globo")

# llamamos al método desplazarse
canela.desplazarse()
nemo.desplazarse()
```

Canela se mueve en 4 patas

Nemo se mueve nadando

¿Qué es, entonces, la POO?

POO, programación orientada a objetos, es un paradigma de programación que busca representar entidades u objetos agrupando datos y métodos que puedan describir sus características y comportamientos.

En este paradigma, los conceptos del mundo real relevantes para nuestro problema a resolver se modelan a través de clases y objetos, y el programa consistirá en una serie de interacciones entre dichos objetos.

¿Qué es un archivo?

Un archivo es una secuencia de datos almacenados en un medio persistente que están disponibles para ser utilizados por un programa. Todos los archivos tienen un nombre y una ubicación dentro del sistema de archivos del sistema operativo.

Un programa no puede manipular los datos de un archivo directamente. Para hacerlo, debe abrir el archivo y asignarlo a una variable, que llamaremos el archivo lógico.



PROCESO

Si el archivo no está previamente creado, se crea al abrirse, esto quiere decir que en el proceso creación y apertura pueden ser una sola etapa, SIEMPRE QUE SE ABRA EN ALGUNO DE LOS MODOS DE ESCRITURA.

Cuando el archivo se crea y abre, se debe especificar para qué se abre: lectura, escritura, agregar contenido al final o lectura + escritura.

```
mi_archivo = open("archivo.txt", "r") # modo lectura
mi_archivo2 = open("archivo2.txt", "w") # modo escritura
mi_archivo3 = open("archivo3.txt", "a") # modo escritura
mi_archivo4 = open("archivo4.txt", "r+") # lectura + escritura
mi_archivo5 = open("archivo5.txt", "w+") # escritura + lectura
```


SIEMPRE QUE SE ABRA UN ARCHIVO, HAY QUE CERRARLO

```
mi_archivo = open("archivo.txt", "r") # modo lectura
mi_archivo2 = open("archivo2.txt", "w") # modo escritura
mi_archivo3 = open("archivo3.txt", "a") # modo escritura
mi_archivo4 = open("archivo4.txt", "r+") # lectura + escritura
mi_archivo5 = open("archivo5.txt", "w+") # escritura + lectura

mi_archivo.close()
mi_archivo2.close()
mi_archivo3.close()
mi_archivo4.close()
mi_archivo5.close()
```

Creación de un archivo

Hay dos formas de crear un archivo, recomiendo se acostumbren a la cláusula `with` porque es la que más utilizan los devs, es como “más profesional”. Además, ayuda a no olvidarse de cerrar el archivo y a trabajar con cada archivo que se utilice de manera ordenada y prolija.

```
with open("archivo6.txt", "w+") as f:  
    f.read()  
    f.close()
```

Si el archivo ya está creado, con `open()` solo se lo agrega a la variable lógica pasándole la ruta de acceso en nuestra computadora como parámetro y desde allí se lo manipula. Si el archivo NO EXISTE en nuestro SO, solo puedo utilizar el método `open()` en los modos `w`, `w+`, `a` y `a+`.

```
with open(r"C:\Users\fravega\Desktop\archivo-prueba.txt", "w+") as mi_archivo3:  
    mi_archivo3.write("Agregando este contenido a mi tercer archivo...")  
mi_archivo3.close()
```

Métodos de manipulación

- `write()`: escribe desde la primera línea del archivo, si éste ya tiene contenido, lo que se pase como parámetro lo sobrescribe. Importante, si el archivo se abre en “a”, cuando se utiliza el método `write()` éste agrega contenido al final del archivo, y agrega ese contenido tantas veces como se le dé run al programa.
- `read()`: lee el contenido del archivo, pero no lo imprime, para poder imprimirlo hay que almacenarlo en una variable e imprimirla.
- `seek()`: ubica el cursor en el índice que se le pase por parámetro.
- `readlines()`: convierte el contenido del archivo en una lista, el separador de cada elemento es el salto de línea, entonces cada elemento es una línea del archivo.
- `tell()`: devuelve la posición en la que se encuentra el cursor.
- `close()`: cierra el archivo.

```
data = '''éste será  
el contenido  
de mi primer archivo'''  
  
with open('archivo.txt', 'w+') as f:  
    f.write(data)  
    f.seek(0)  
    print(f.read())  
    print(f.tell())
```

¿Qué es un módulo?

Un módulo es un tipo de archivo `.py` o `.pyc` que contiene funciones, clases, objetos, variables e incluso otros módulos para ser utilizados en otros archivos de Python.

Se utiliza para que el código sea más ordenado, limpio, legible y reutilizable.

Para poder usar el contenido de un módulo es necesario importarlo en el archivo en el que haremos uso del mismo.

```
# Siempre se importan los módulos y paquetes al inicio
```

```
import funciones_mate
```

```
# Utilizo el contenido:
```

```
suma = funciones_mate.sumar(1, 2)
```

```
# Siempre se importan los módulos y paquetes al inicio
```

```
import funciones_mate as fm
```

```
# Utilizo el contenido:
```

```
suma = fm.sumar(1, 2)
```

```
# Siempre se importan los módulos y paquetes al inicio
```

```
from funciones_mate import sumar, restar
```

```
suma = sumar(1, 2)
```

```
resta = restar(3, 4)
```

```
# Siempre se importan los módulos y paquetes al inicio
```

```
from funciones_mate import *
```

```
suma = sumar(1, 3)
```

```
resta = restar(4, 5)
```

```
multiplicacion = producto(6, 7)
```

```
division = dividir(8, 4)
```

```
potenciacion = potencia(3, 5)
```


¿Qué es un paquete?

Un paquete es un directorio o carpeta donde se almacenan diferentes módulos que están relacionados entre sí.

Se crean con una carpeta que tenga sí o sí un archivo (file) llamado `__init__.py`.
Dentro de un paquete puede haber otros paquetes, es importante que cada paquete interno tenga su propio file `__init__.py`.

Para importar es similar a los módulos.

¿Qué es una excepción?

Una excepción es un error que ocurre durante la ejecución de un programa pero que no necesariamente está en la sintaxis del código.

El ejemplo más común es la división por 0, si bien la sintaxis es correcta, el programa lanza un error porque, como sabemos, no se puede dividir por 0.

```
# Funciones matemáticas

def sumar(a, b):
    return a + b

def restar(a, b):
    return a - b

def multiplicar(a, b):
    return a * b

def divifir(a, b):
    return a / b

num1 = 5
num2 = 0

print(sumar(num1, num2))
print(restar(num1, num2))
print(multiplicar(num1, num2))
print(divifir(num1, num2))
```



```
5
5
0
Traceback (most recent call last):
  File "excepciones.py", line 23, in <module>
    print(divifir(num1, num2))
  File "excepciones.py", line 14, in divifir
    return a / b
ZeroDivisionError: division by zero
```

Estos errores se pueden incluir en nuestra sintaxis, de esta manera se evita que el programa se rompa y pueda seguir su curso normalmente.

¿Cómo?

- ➔ Primero debemos saber el nombre del error, en este caso ZeroDivisionError.
- ➔ Segundo, debemos incluir la sentencia try-except en nuestro código, para poder contemplar este error.

```
def dividir(a, b):  
    try:  
        return a / b  
    except ZeroDivisionError:  
        return f"No se puede dividir por cero. Operación fallida"
```

Aplicado try-except, lo que hacemos es indicar que, si está todo ok se efectúa el cálculo, y si aparece el error de ZeroDivisionError se devuelve el string de aviso.

Al ser ejecutada la función dividir(), se intentará realizar excepto que el parámetro b sea 0, y devolverá ese mensaje.

```
5
5
0
No se puede dividir por cero. Operación fallida
```

También se puede agregar la sentencia `finally` que no es obligatoria, pero siempre que esté se ejecutará, haya o no haya error.

```
def dividir(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return f"No se puede dividir por cero. Operación fallida"
    finally:
        return "El cálculo ha finalizado"
```

Algunos ejemplos de las más comunes:

- **ValueError**: cuando se espera un valor y se recibe otro.
- **NameError**: cuando se recibe el nombre de una variable, función, clase u objeto pero no fue previamente definido.
- **TypeError**: cuando el tipo de dato que se quiere manipular debería ser otro.

¿Qué es una expresión regular (Regex)?

Una expresión regular es una secuencia de caracteres que forma un patrón que sirve para realizar búsquedas y filtrar datos.

Para poder utilizar las regex, hay que importar la librería re:

```
import re
```

Hay muchísimas regex, vamos a ver las más utilizadas, pero para saber el listado total y cómo trabajar con cada una pueden ingresar a este link

<https://docs.python.org/3/library/re.html>

```
import re

texto = "Mi nombre es Magalí y tengo 28 años"
buscar = "nombre"

if re.search(buscar, texto):
    print(f"La cadena contiene {buscar} entre sus caracteres")
else:
    print(f"La cadena no contiene {buscar} entre sus caracteres")

texto_encontrado = re.search(buscar, texto)
print(texto_encontrado.start()) # me devolverá la posición del primer caracter de la cadena donde se encuentra
print(texto_encontrado.end()) # me devovlerá la posición del caracter siguiente al último de donde se encuentra
print(texto_encontrado.span()) # devuelve una tupla de dos números, el primero es start() y el segundo end()

cadena = "Me gusta la programación, estudié programación, trabajo en programación"
buscar = "programación"

print(re.findall(buscar, cadena)) # devolverá una lista con la cantidad de veces que aparece el patrón en la cadena
print(len(re.findall(buscar, cadena))) # devolverá la cantidad de veces que aparece el patrón en la cadena
```

Como pueden observar, la sintaxis es:

re.método(patrón, variable)

Metacaracteres

Se conoce como metacaracteres a aquellos que, dependiendo del contexto, tienen un significado especial para las expresiones regulares.

Algunos autores los llaman “caracteres comodín”.

Tipos

- Anclas
- Clases de caracteres
- Rangos
- Universal
- Cuantificadores

Metacaracteres – anclas

Indican que lo que queremos encontrar se encuentra al principio o al final de la cadena. Combinándolas, podemos buscar algo que represente a la cadena entera.

- **^patrón**: coincide con cualquier cadena que comience con patrón.
- **patrón\$**: coincide con cualquier cadena que termine con patrón.

```
patron = "hola$"  
  
cadena = "buenos días, hola"  
  
if re.search(patron, cadena):  
    print(True)  
  
patron_2 = "^mi nombre"  
cadena_2 = "mi nombre es Magali"  
  
if re.search(patron_2, cadena_2):  
    print(True)
```

Metacaracteres - clases

Se utilizan cuando se quiere buscar un caracter dentro de varias posibles opciones. Una clase se delimita entre corchetes y lista posibles opciones para el caracter que representa.

- `[abc]`: coincide con a, b, o c
- `[387ab]`: coincide con 3, 8, 7, a o b
- `niñ[oa]s`: coincide con niños o niñas.
- Para evitar errores, en caso de que queramos crear una clase de caracteres que contenga un corchete, debemos escribir una barra \ delante, para que el motor de expresiones regulares lo considere un caracter normal: la clase `[ab\[]` coincide con a, b y [.

```
import re
```

```
patron = '[AR]'
```

```
lista_vuelos = [
```

```
    'AR2607',
```

```
    'AR2606',
```

```
    'AA990',
```

```
    'AA991',
```

```
    'IB894',
```

```
    'LA789'
```

```
]
```

```
for vuelo in lista_vuelos:
```

```
    if re.search(patron, vuelo):
```

```
        print(vuelo)
```

Metacaracteres - rangos

Si queremos encontrar un número, podemos usar una clase como `[0123456789]`, o podemos utilizar un rango. Un rango es una clase de caracteres abreviada que se crea escribiendo el primer caracter del rango, un guión y el último caracter del rango. Múltiples rangos pueden definirse en la misma clase de caracteres.

- `[a-c]`: equivale a `[abc]`
- `[0-9]`: equivale a `[0123456789]`
- `[a-d5-8]`: equivale a `[abcd5678]`
- Es importante aclarar que si se quiere buscar un guión debe colocarse al principio o al final de la clase:
`[a4-]` `[-a4]` `[a\-4]`


```
import re

patron = 'AR[1][1-7][0-9][0-9]'

lista_vuelos = [
    'AR1133',
    'AR1120',
    'AR2606',
    'AR2607',
    'AR1879'
]

for vuelo in lista_vuelos:
    if re.search(patron, vuelo):
        print(f'El vuelo {vuelo} es internacional')
    else:
        print(f'El vuelo {vuelo} es de cabotaje')
```

Para practicar:

www.regex101.com

Unidad II

INTRODUCCIÓN A LA CIENCIA DE DATOS



¿Qué es la ciencia de datos?



La ciencia de datos es un **campo interdisciplinario** que utiliza métodos, procesos, algoritmos y sistemas científicos para extraer valor de los datos. Los científicos de datos combinan una variedad de habilidades, entre ellas estadísticas, informática y conocimiento empresarial, para analizar datos recopilados de la web, de teléfonos inteligentes, de clientes, sensores y otras fuentes. La ciencia de datos revela tendencias y genera información que las empresas pueden utilizar para tomar mejores decisiones y crear productos y servicios más innovadores. **Los datos son el cimiento de la innovación**, pero su valor proviene de la información que los científicos pueden extraer y luego utilizar a partir de los mismos.

¿Qué es la data?

Data se considera a la información almacenada en algún formato digital que luego se puede utilizar como base para análisis y tomar decisiones.

Hay dos tipos de data: t

- traditional data
- big data

Enfrentarse a la data es el primer paso para resolver un problema de cualquier tipo.

TRADITIONAL DATA: datos estructurados en tablas (texto y número) y que pueden ser manejados desde una sola computadora.

BIG DATA: volúmenes de datos extremadamente grandes no solo en términos de volumen y que poseen varios formatos a la vez (estructurado, semi-estructurado o desestructurado). 3Vs: volumen, variedad, velocidad.

DATA

DATOS PROCESADOS

INFORMACIÓN



TD

Lenguajes y programación

- Python
- R
- Matlab
- SQL (bases de datos)

Software

- Excel
- SPSS
- SAP

BG

Lenguajes y programación

- Python
- R
- Java
- Scala

Software

- Hadoop
- Apache
- MongoDB

Puestos de trabajo

DATA ARCHITECT

DATA ENGINEER

**DATABASE
ADMINISTRATOR**

**BIG DATA
ARCHITECT**

**BIG DATA
ENGINEER**

Antes de continuar...

¿saben la diferencia entre Data
Analysis y Data Analytics?



Entonces, el primer paso sería **identificar qué volumen de datos** tenemos, para saber si vamos a trabajar con **TD** o **BD**. Si bien son distintas, ambas cumplen el mismo rol » materia prima del análisis de datos.

Lo que se conoce como Data-Driven decisions requieren que esos datos estén ordenados y sean relevantes para nuestro trabajo.

Una vez que sabemos con qué tipo de data vamos a lidiar, pasamos al a parte de la ciencia de datos que implica:

- Business Intelligence (analysis)
- Traditional Methods (analytics – predicciones)
- Machine Learning (analytics – predicciones)

¿Qué es la Inteligencia de Negocios (BI)?

Business Intelligence es la parte de la ciencia de datos que incluye las tecnologías involucradas en el proceso de analizar, entender y reportar (dashboards & reports) data disponible del pasado, para guiar el camino estratégico hacia la toma de decisiones empresariales, extrayendo ideas y encontrando patrones.

Una vez que se hace el análisis y se extrae la información confiable y requerida, ésta es presentada en tres tipos de formato:

- Dashboards: gráficos
- Reports: prosa
- Dashboards + reports: combinación

- Ayuda a entender cómo ha crecido nuestro negocio y porqué, o porqué no.
- Cómo le está yendo a nuestros competidores en relación a nosotros.
- Explica las variaciones en los precios de los proveedores.
- Muestra qué tan bueno o malo fue nuestro año en términos económicos y financieros.



Entonces, forma parte del proceso de análisis porque se encarga de hechos que ya sucedieron y de los que se puede recabar información para la posterior toma de decisiones.

Lenguajes y programación

- Python
- R
- Matlab
- SQL (bases de datos)

Software

- Excel
- PowerBI
- Tableau
- QlikSense

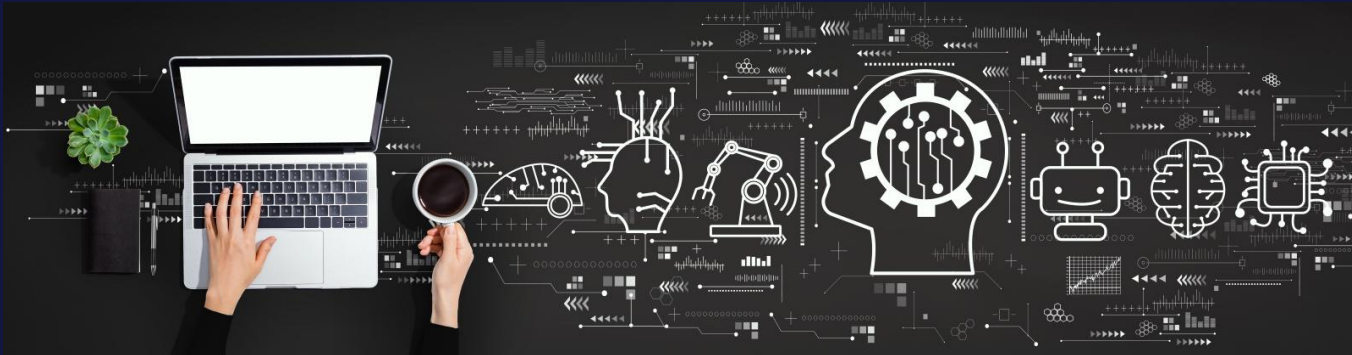
Puestos de trabajo



Predictive Analytics

Luego de que se ha hecho el análisis de BI y se tienen los dashboards y reports correspondientes, se procede a seguir procesando dicha información para generar predicciones a futuro a través de

- Traditional methods
- Machine Learning



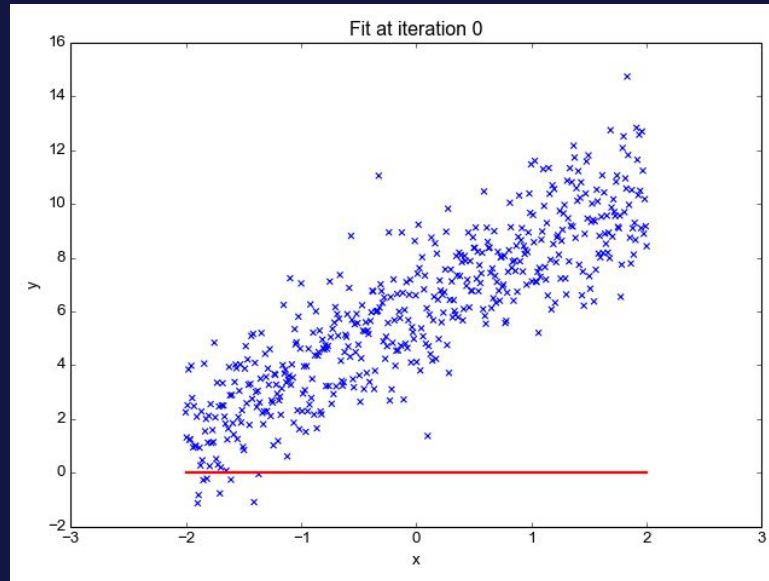
1. Traditional Methods

Se evalúan posibles escenarios futuros a través de la utilización de métodos estadísticos avanzados.



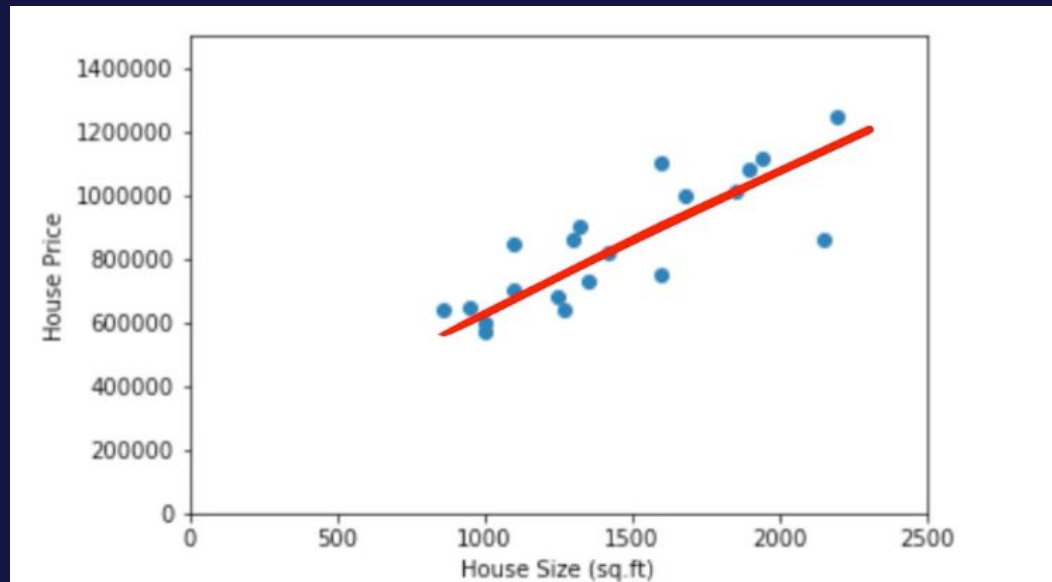
Traditional Methods - Linear Regression

Es un modelo utilizado para cuantificar las relaciones causales entre las diferentes variables incluidas en nuestro análisis.



	House Price (\$)	House Size (sq.ft.)
0	1116000	1940
1	860000	1300
2	818400	1420
3	1000000	1680
4	640000	1270
5	1010000	1850
6	600000	1000
7	700000	1100
8	1100000	1600
9	570000	1000
10	860000	2150
11	1085000	1900
12	1250000	2200

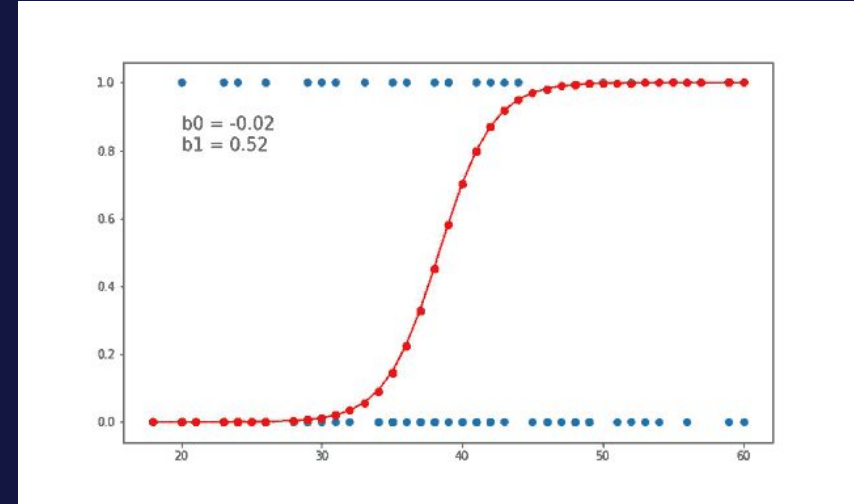
$$y=bx$$



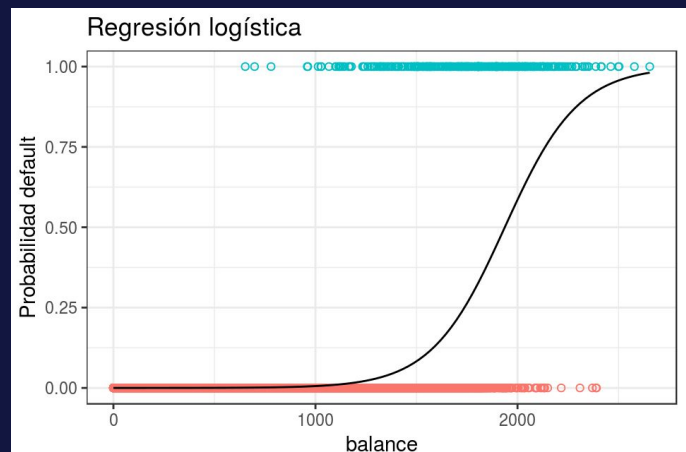
Traditional Methods – Logistic Regression

Es un modelo utilizado para poder analizar variables categóricas y darles así dos valores lógicos: cero o 1 (True o False). Lo que hacemos es convertir una variable cualitativa en una cuantitativa y tratarla como tal.

Un ejemplo común es la elección de talentos en una empresa.

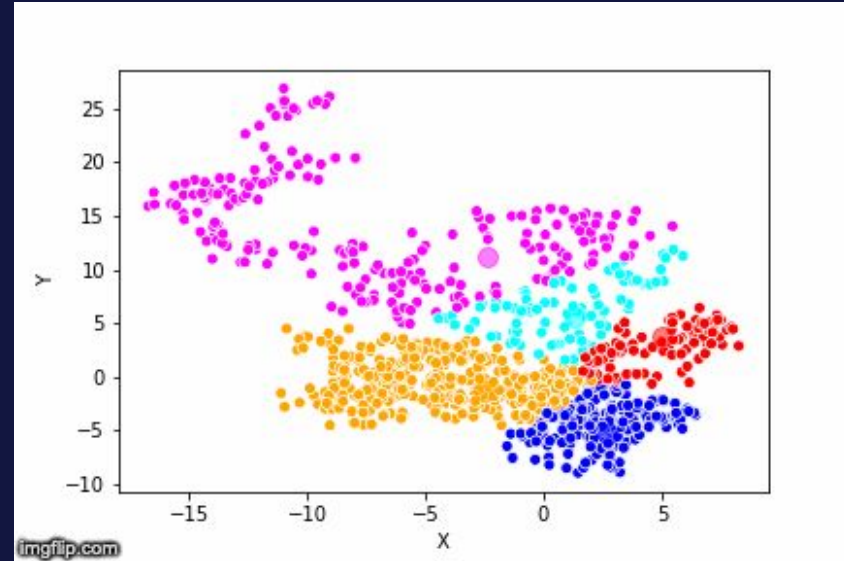


	default <dbl>	balance <dbl>
1	0	729.5265
2	0	817.1804
3	0	1073.5492
4	0	529.2506
5	0	785.6559
6	0	919.5885
6 rows		





Traditional Methods - Clustering

Es un modelo utilizado mediante la agrupación de factores comunes. En el ejemplo de las viviendas, se podrían agrupar por barrios y clusterizar, y así notar cómo varía el precio de una casa según su ubicación (ejemplo: CABA).



Traditional Methods - Clustering

Una misma ecuación, puede tener muchísimas variables exploratorias, pero podemos identificar cuáles pertenecen a un mismo tópico y agruparlas.

$$y = \underbrace{b_1x_1 + b_2x_2 + b_3x_3}_{b_nx_n} + b_4x_4 + b_5x_5 + b_6x_6 + \dots +$$


Agrupación: **factor analysis**

$$x_1 + x_2 + x_3 = Z$$

cluster #1

cluster #2

factor #1

factor #2

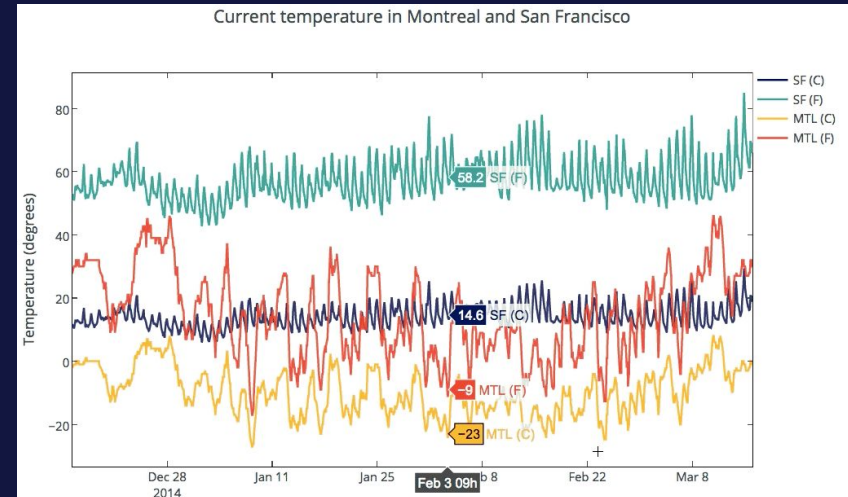
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
Observation 1															
Observation 2															
Observation 3															
Observation 4															
Observation 5															
Observation 6															
Observation 7															
Observation 8															
Observation 9															

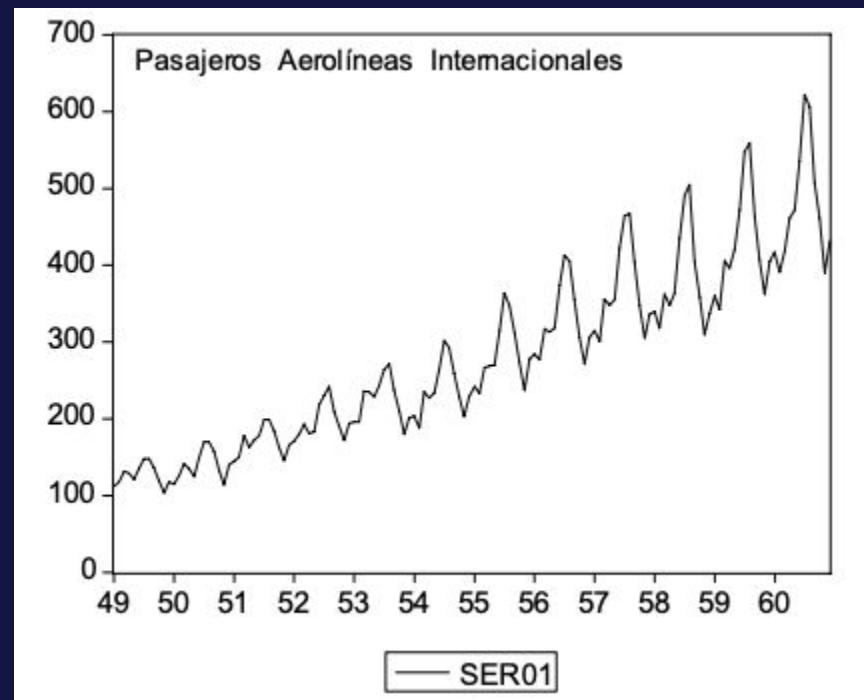
Clustering != Factor analysis

- Clustering: agrupar las observaciones
- Factor analysis: agrupar los factores

Traditional Methods - Time Series

Es un modelo donde los datos estadísticos se recopilan, observan o registran en intervalos de tiempo regulares (diario, semanal, semestral, anual, entre otros). El término serie de tiempo se aplica por ejemplo a datos registrados en forma periódica que muestran, por ejemplo, las ventas anuales totales de almacenes, el valor trimestral total de contratos de construcción otorgados, el valor anual del PBI, etc. **La variable independiente siempre es el tiempo.**





Lenguajes y programación

- Python
- R
- Matlab

Software

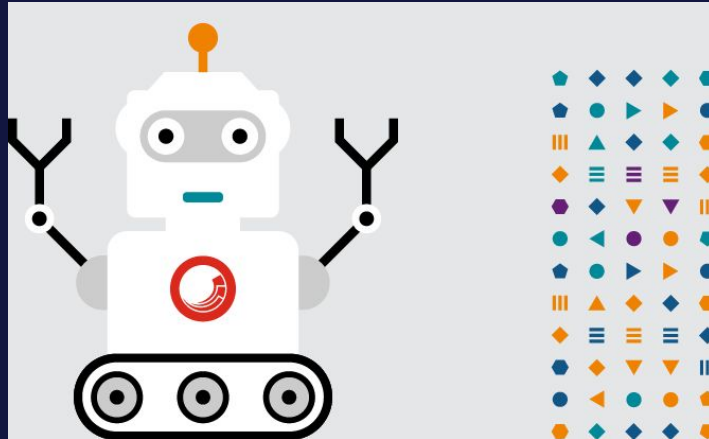
- Excel
- SPSS
- IBM
- STATA

Puestos de trabajo



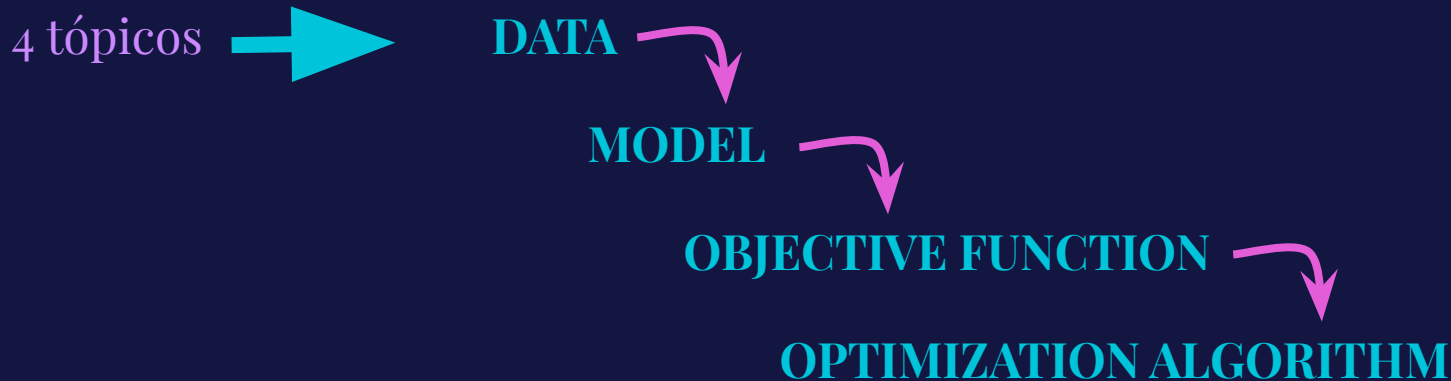
2. Machine Learning

El aprendizaje automático consiste en crear un algoritmo que la computadora usa para encontrar un modelo que se ajuste a los datos de la mejor manera posible, y hace predicciones muy específicas basada en ellos. Le “damos” a la computadora un algoritmo para que “aprenda” sola.

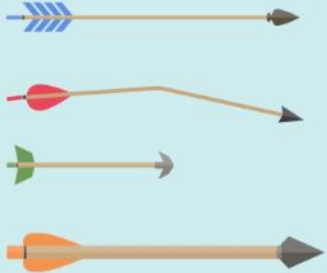


¿Qué es, entonces, un Machine Learning Algorithm?

Un algoritmo de ML es un proceso de prueba y error, con la característica -fundamental- de que cada nuevo intento es AL MENOS tan bueno como el anterior, es decir, que se va perfeccionando en cada prueba.



Data



Model

the usage of
the bow



**Objective
Function**

calculate how far
from the target



**Optimization
algorithm**

mechanics that will
improve the model's
performance



Al final del proceso, el robot (model) habrá entrenado el uso de las flechas (datos) y habrá logrado lanzarlas al medio (objective function) o minimizado el desvío. Cuando ya no se pueda mejorar nada (optimization algorithm) el proceso termina.

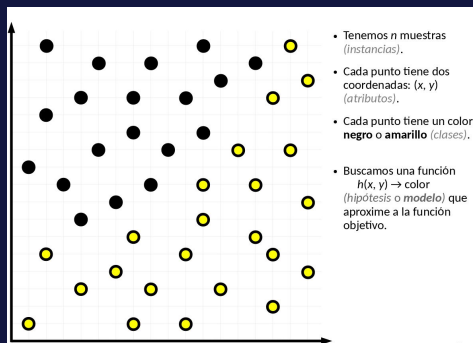
**“En el corto o mediano plazo, no
seremos reemplazados por robots ni
por computadoras, pero sí por
personas que sepan utilizarlos.”**



Machine Learning - Aprendizaje Supervisado

Aquel en el que entrenar el algoritmo de ML es similar a cuando un docente supervisa una clase (por ejemplo: el robot con el arco y las flechas de recién).

Aproximamos la **función objetivo** construyendo un **modelo** de Machine Learning.

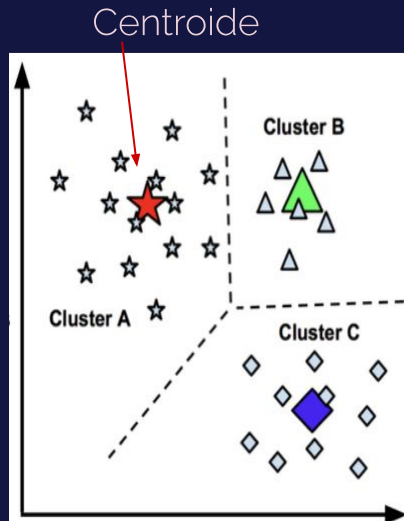


Entrenar un modelo consiste en **ajustar sus parámetros** o encontrar sus valores óptimos dado un conjunto de datos.

Algoritmos de aprendizaje automático:
procedimientos para entrenar modelos

Machine Learning - Aprendizaje No Supervisado

Aquel en el que entrenar el algoritmo de ML nuestros datos no están etiquetados, es decir, son valores dispares y desconocidos. Entonces es necesario que el modelo divida los datos de cierta forma, para aproximarse a la función objetivo.



CLUSTER: k-means

Separa los datos en k clusters ubicando a las instancias que estén dentro de una región cercana dentro de un mismo cluster.

Encuentra un número k de centroides, uno por cada cluster, tal que la distancia entre los centroides y los datos más cercanos sea la mínima posible.

Entonces...

El Machine Learning es una rama de la Inteligencia Artificial que se dedica al estudio de los algoritmos que aprenden a realizar una tarea en base a la experiencia.

NOTA: los modelos pueden ser híbridos, es decir, tener tanto supervised learning como unsupervised learning algorithms.

Machine Learning - Reinforcement Learning

Aprendizaje automático basado en “premios”. Une los dos tipos anteriores. El algoritmo de aprendizaje recibe una valoración sobre la relevancia de la respuesta dada. Si la respuesta es correcta, el aprendizaje por refuerzo actúa como el aprendizaje supervisado, en ambos casos el aprendiz recibe información de lo que es apropiado. Sin embargo, ante las respuestas erróneas ambas aproximaciones difieren significativamente cuando el aprendiz responde de forma inadecuada. De este modo, el aprendizaje supervisado le dice exactamente al aprendiz qué debería haber respondido, mientras que **el aprendizaje por refuerzo solo le informa acerca de que el comportamiento ha sido inapropiado y (normalmente) cuánto error se ha cometido**. La aproximación del aprendizaje por refuerzo, es más habitual en la naturaleza que en el aprendizaje supervisado.

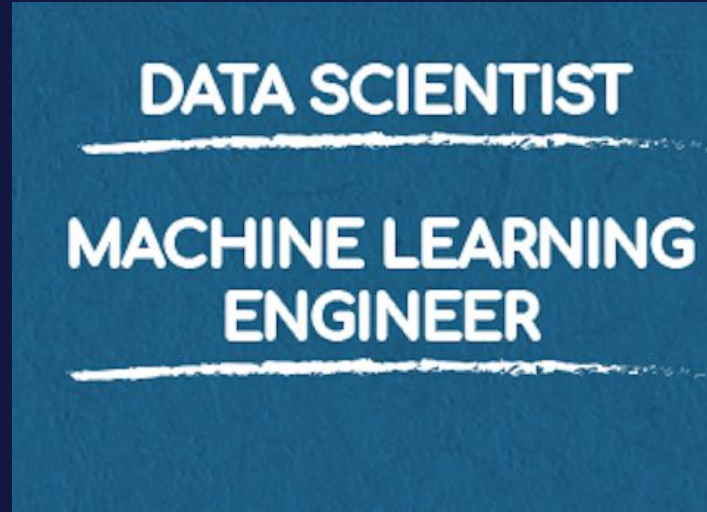
Lenguajes y programación

- Python
- R
- Matlab
- Java
- Javascript
- C
- C++
- Scala

Software

- Microsoft Azure
- Rapidminer

Puestos de trabajo



Unidad III

ESTADÍSTICA



Medidas de centralización

Las medidas de centralización son aquellas que nos indican **cómo se sitúan los datos**: media, mediana, percentiles y moda.



MEDIA: la media es *el valor promedio de un conjunto de datos numéricos*, calculada como la suma del conjunto de valores dividida entre el número total de valores.

MEDIANA: representa *el valor de la variable de posición central en un conjunto de datos ordenados*. Si la cantidad de datos es par, se toma el promedio de los dos datos centrales.

PERCENTILES: *implica la división del total de aquello que se está midiendo en 99 partes para obtener un total de 100 partes iguales*. De esta manera, la totalidad está representado en algún lugar de estas 99 partes, y el o los datos concretos ocuparán una posición entre dichas partes. Se trata de un tipo de cuantil o fractil, valores que permiten separar datos en grupos con el mismo número de valores.

MODA: la moda es *el valor con mayor frecuencia en una de las distribuciones de datos*.

```
import numpy as np
from scipy import stats

lista_numeros = [100, 9, -5, -5, 6, 99, 200, 44, 21, 789, -10, -11, 21, 30]

arr_numeros = np.array(lista_numeros) # creo el arreglo a partir de la lista

# Media

media = np.mean(arr_numeros)
print(media)

# Mediana

mediana = np.median(arr_numeros)
print(mediana)

# Percentiles

cuartil = np.percentile(arr_numeros, 25)
print(cuartil)

# Moda

moda = stats.mode(arr_numeros)
print(moda)
```

92.0

21.0

-2.25

ModeResult(mode=array([-5]), count=array([2]))

Medidas de dispersión

Las medidas de dispersión son parámetros estadísticos que *indican cómo se alejan los datos respecto de la media aritmética y sirven como indicadores de la variabilidad de los datos*: varianza y desvío estándar.

Esperanza matemática	$\mu = \sum_{i=1}^n x_i \cdot p_i$	(Media)
Varianza	$\sigma^2 = \sum_{i=1}^n x_i^2 \cdot p_i - \mu^2 = \sum_{i=1}^n (x_i - \mu)^2 \cdot p_i$	
Desviación típica	$\sigma = + \sqrt{\sum_{i=1}^n x_i^2 \cdot p_i - \mu^2}$	

VARIANZA: es una medida de dispersión definida como *la esperanza del cuadrado de la desviación de dicha variable respecto a su media*.

DESVÍO ESTÁNDAR: es la medida de dispersión más común, que *indica qué tan dispersos están los datos con respecto a la media*. Es la raíz cuadrada de la varianza.

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n-1} \quad s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}}$$


```
# Varianza
```

```
varianza = np.var(arr_numeros)  
print(varianza)
```

```
# Desvío estándar
```

```
desvio_e = np.std(arr_numeros)  
print(desvio_e)
```

40566.57

201.41