



machine learning in Python



MACHINE LEARNING/ APRENDIZAJE AUTOMÁTICO

Sergio Hernán Valenzuela Cámara

serginho61@gmail.com

sergio@tecnopolis.ai

OptionSoft

2018-2019

INTRODUCCIÓN A LA CLASIFICACION/ CONTENIDO

Introducción

Clasificando emails, animales y mucho mas

Importando, clasificando y validando un modelo

Clasificación de variables categóricas

El problema del éxito y el algoritmo tonto

Naïve Bayes y maximum a posteriori por dentro

Probando diferentes modelos y validando el vencedor

Nuevos conceptos de clasificación

Utilizando K-Fold

Creando un diccionario

Clasificando los textos y ganando productividad en la empresa

Quebrando en la puntuación adecuada

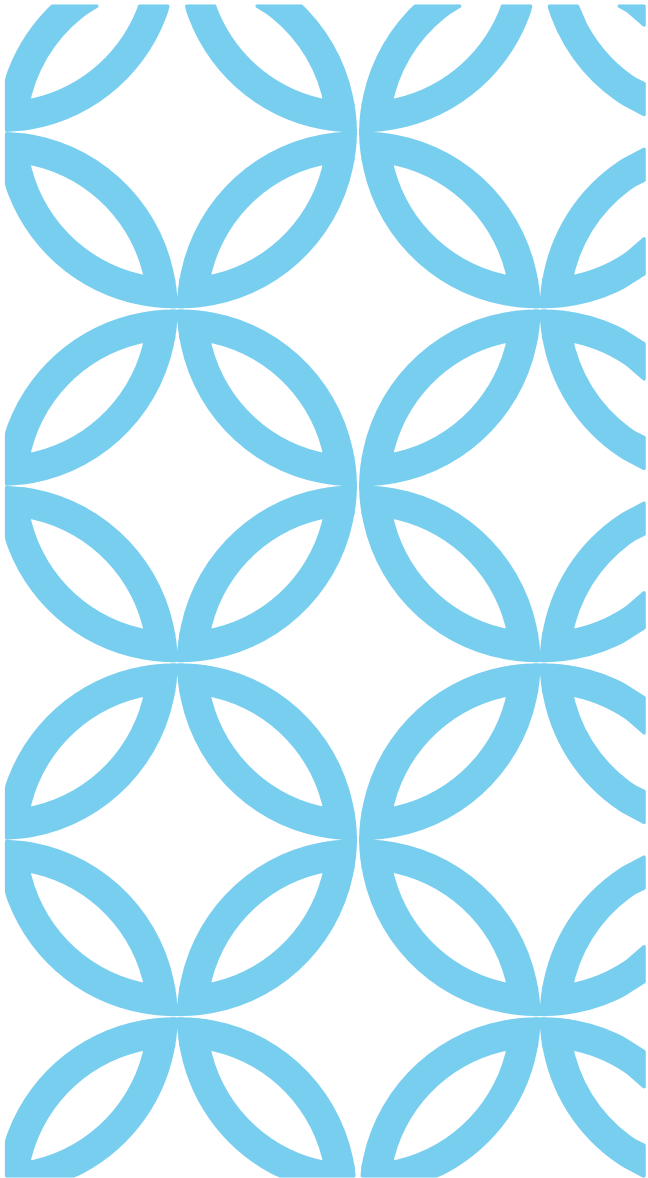
Conclusión

Bibliografía

INTRODUCCIÓN

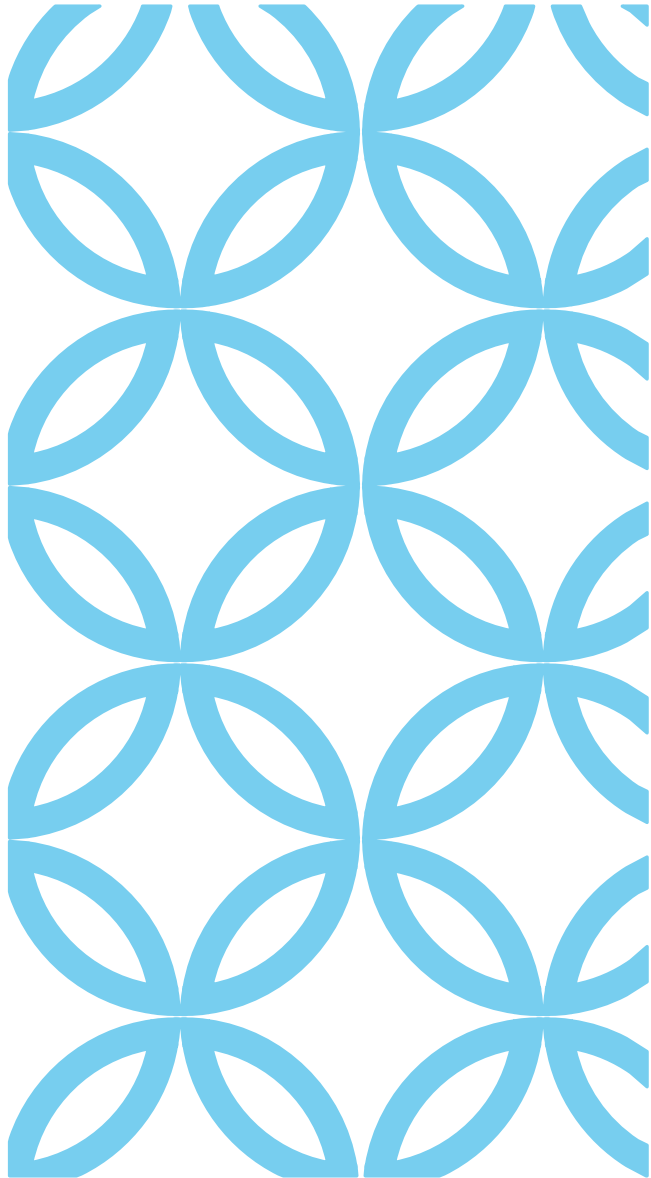
¿ Qué es Machine Learning ?





El aprendizaje automático o aprendizaje automatizado o aprendizaje de máquinas es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos.

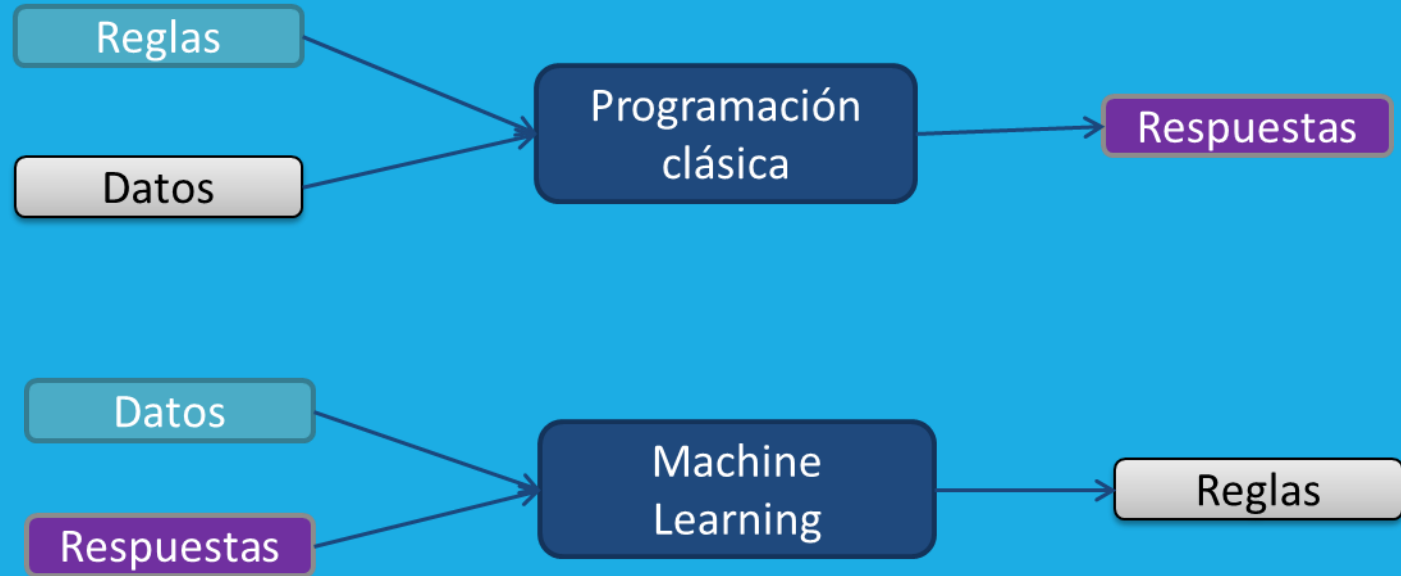
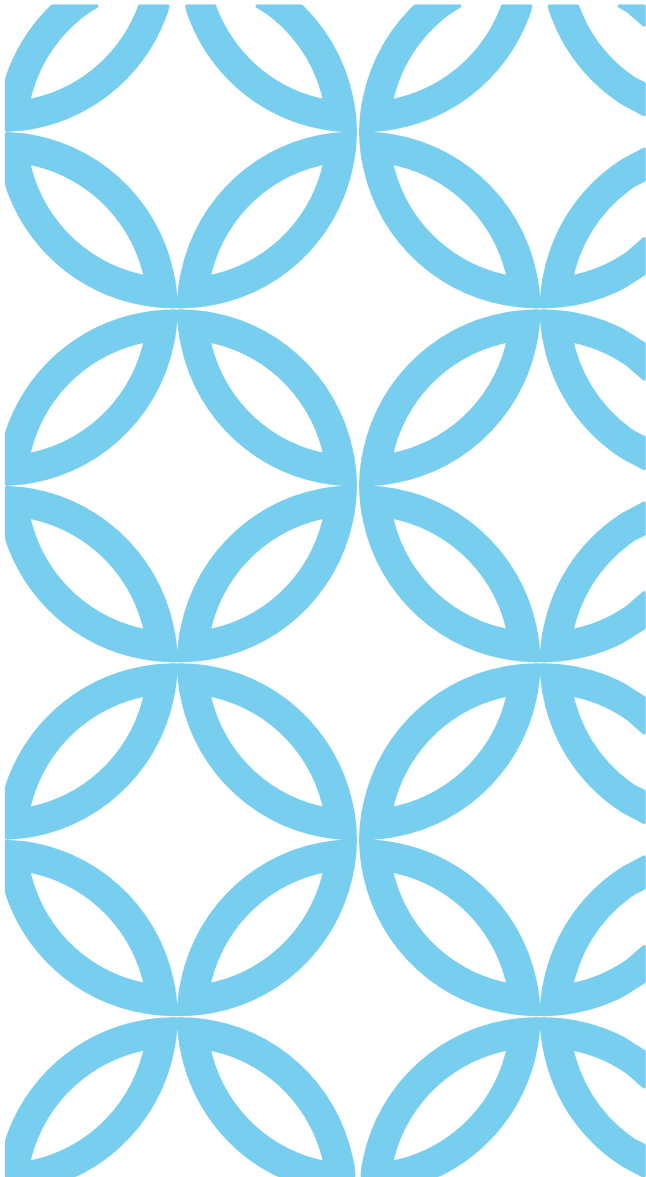
MACHINE LEARNING (APRENDIZAJE AUTOMÁTICO)



- Aprender a través de los datos
- No hay una programación explícita
- Descubrimiento de patrones ocultos
- Toma de decisiones basada en los datos



MACHINE LEARNING (APRENDIZAJE AUTOMÁTICO) #2

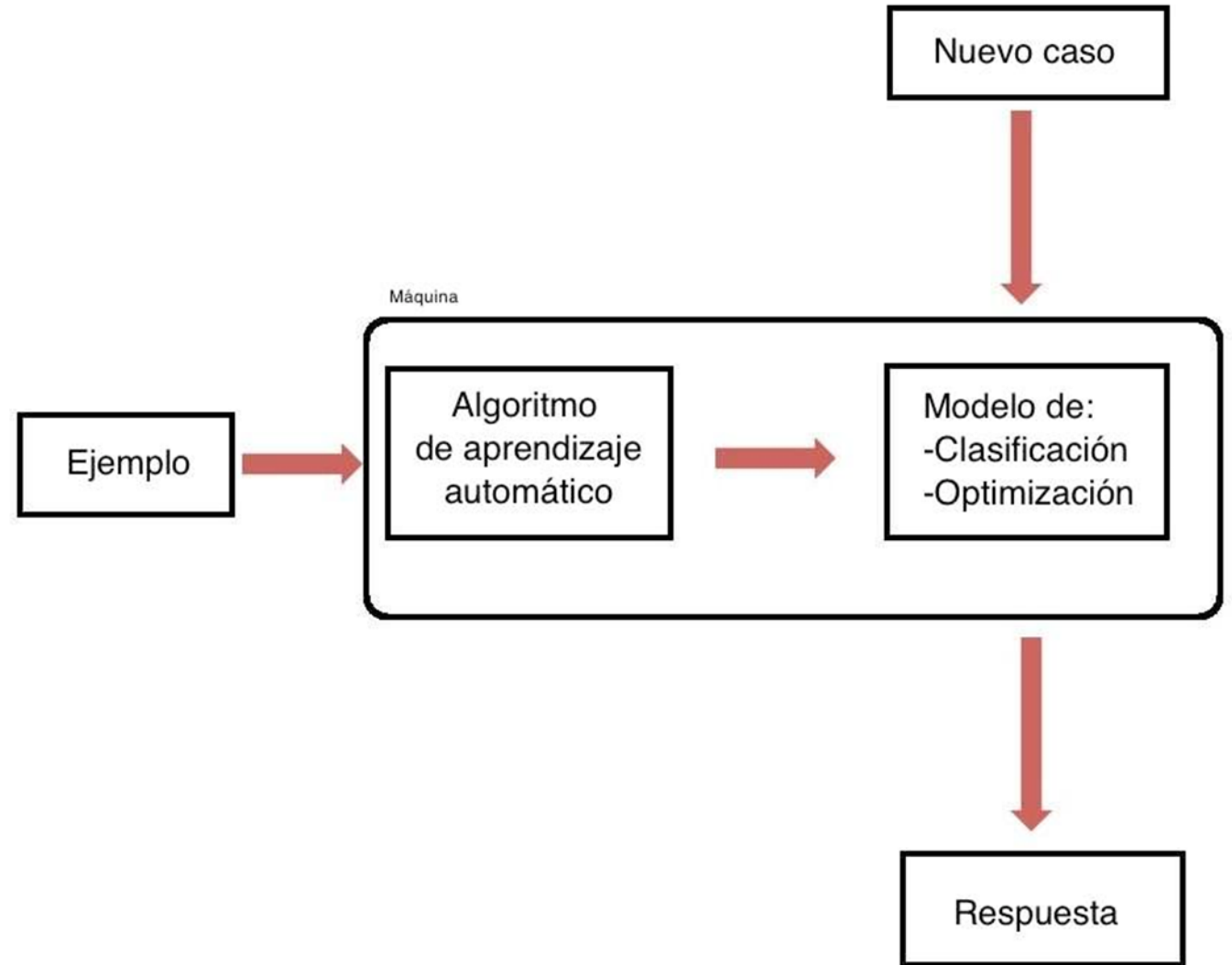


MACHINE LEARNING EL NUEVO PARADIGMA EN LA PROGRAMACIÓN

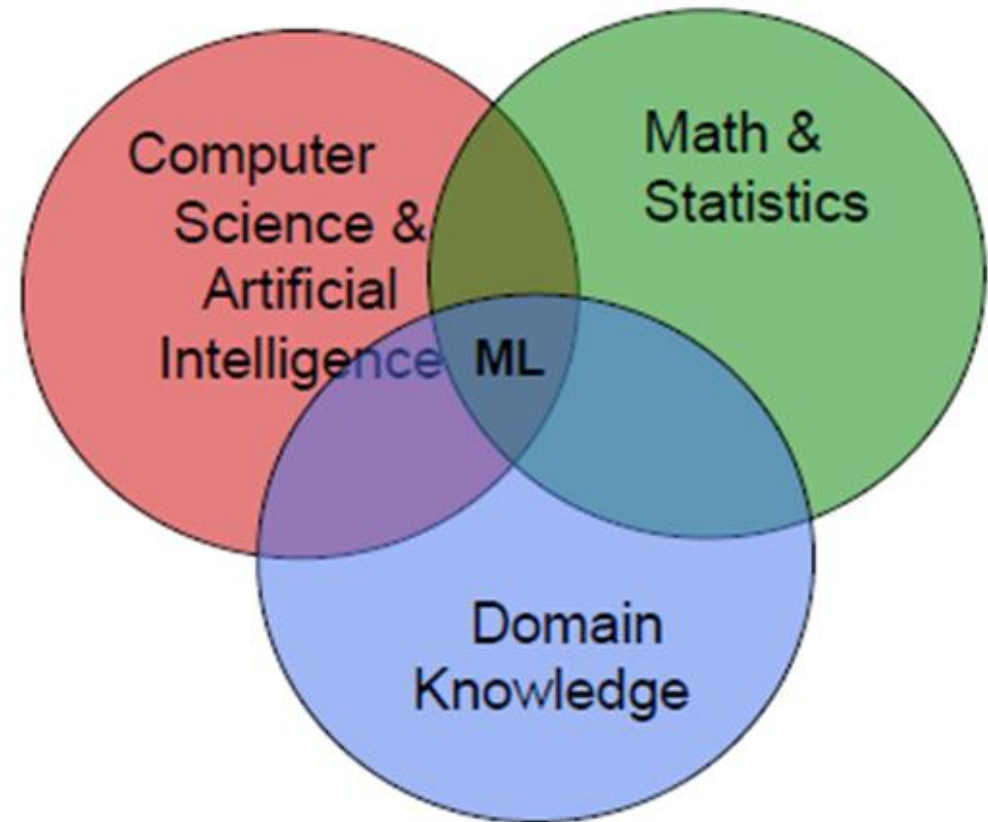
Ejemplos

Emails: clasificar Spam o No Spam

Animales: clasificar si es perro o
cerdo

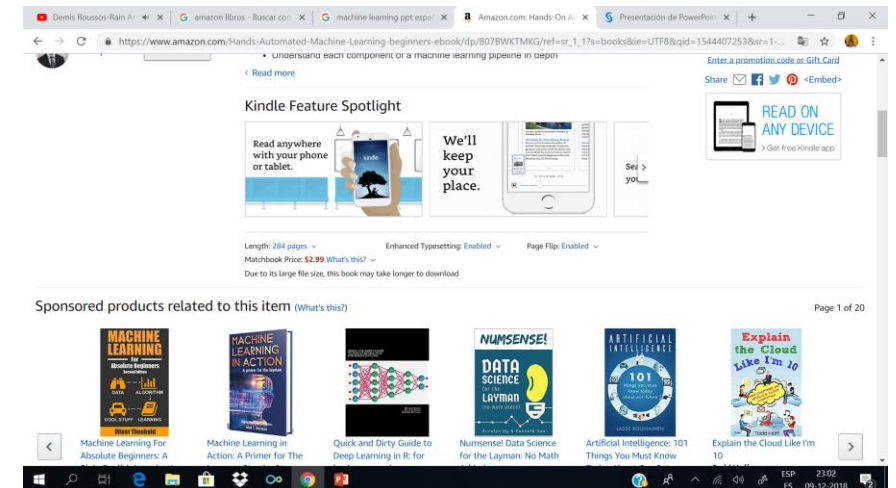


MACHINE LEARNING IS MULTIDISCIPLINAR



¿ EN QUÉ PODEMOS APLICAR MACHINE LEARNING ?

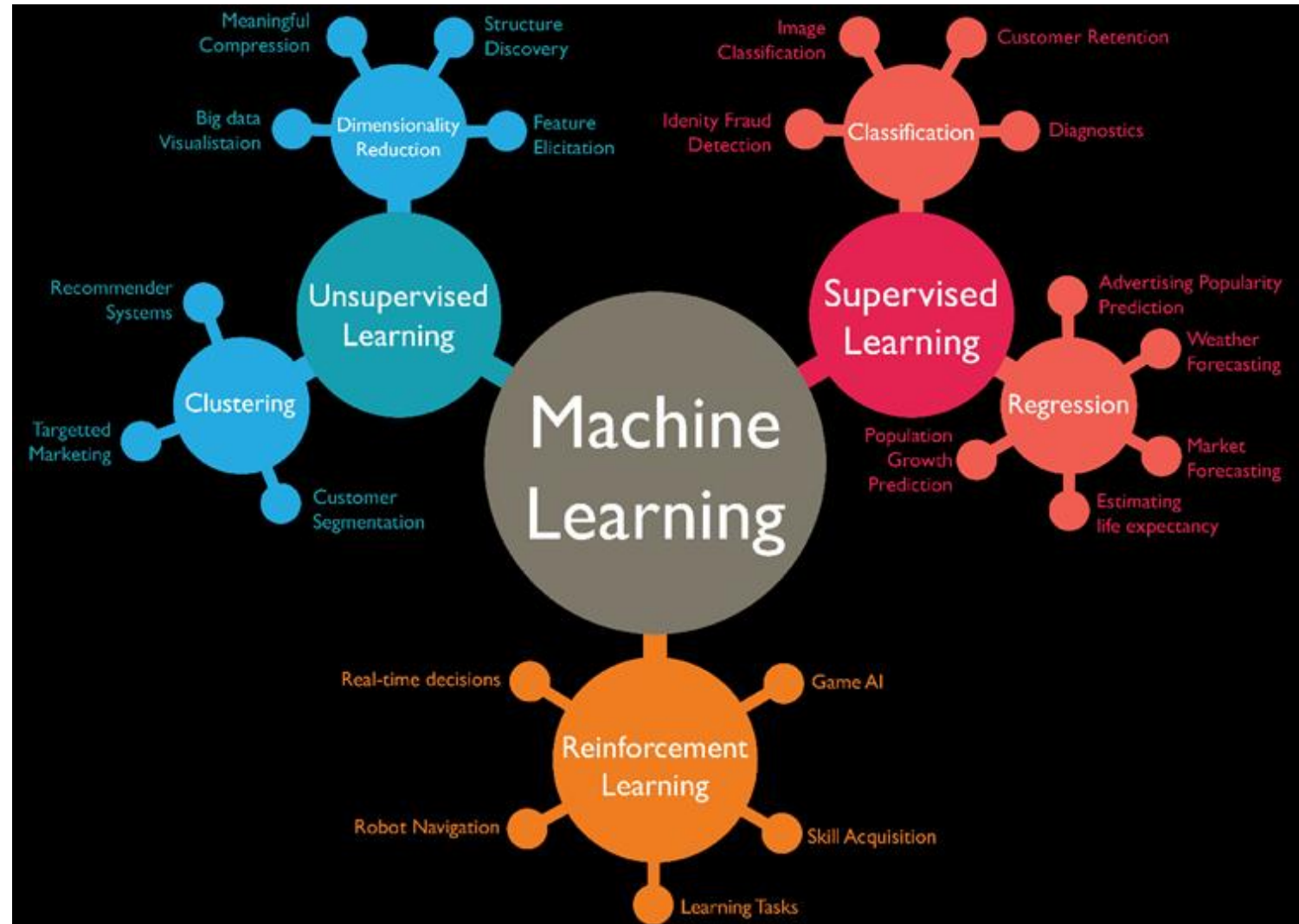
- Detección de fraudes en tarjetas de crédito
- Recomendaciones de compra (Netflix: películas; Amazon: libros)
- Anuncios orientados en aplicaciones móviles
- Análisis de sentimiento en las redes sociales
- Monitorización climática para detectar patrones estacionales
- Detección de patrones en la lucha contra el crimen
- Etc,etc,etc.



CATEGORÍAS DEL MACHINE LEARNING

- Clasificación
- Regresión
- Segmentación (Clustering)
- Asociación

CATEGORÍAS DEL MACHINE LEARNING



Objetivo: Predecir una categoría



Soleado

Ventoso

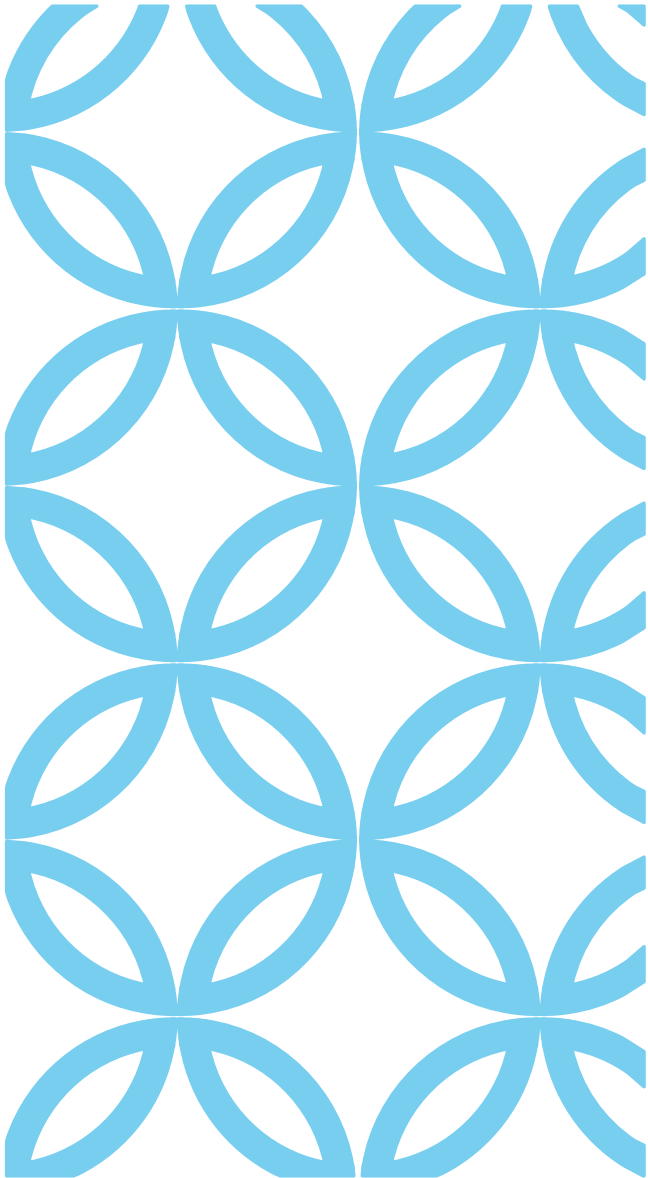
Lluvioso

Nublado

CLASIFICACIÓN

CONCEPTOS

¿Qué es una feature?



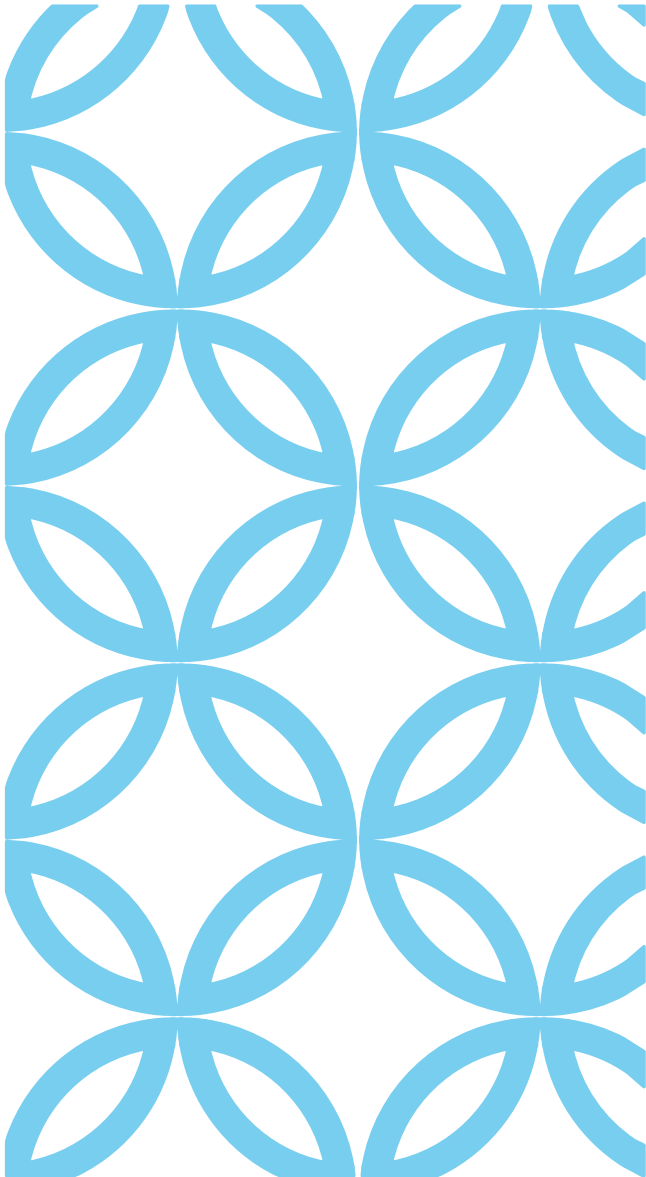
Una feature (característica) en general, son las medidas numéricas individuales que utilizamos para describir nuestras características (features) de datos.

Estas características se pueden medir y calcular directamente a partir de datos intermedios”

FEATURE (CARACTERÍSTICA)

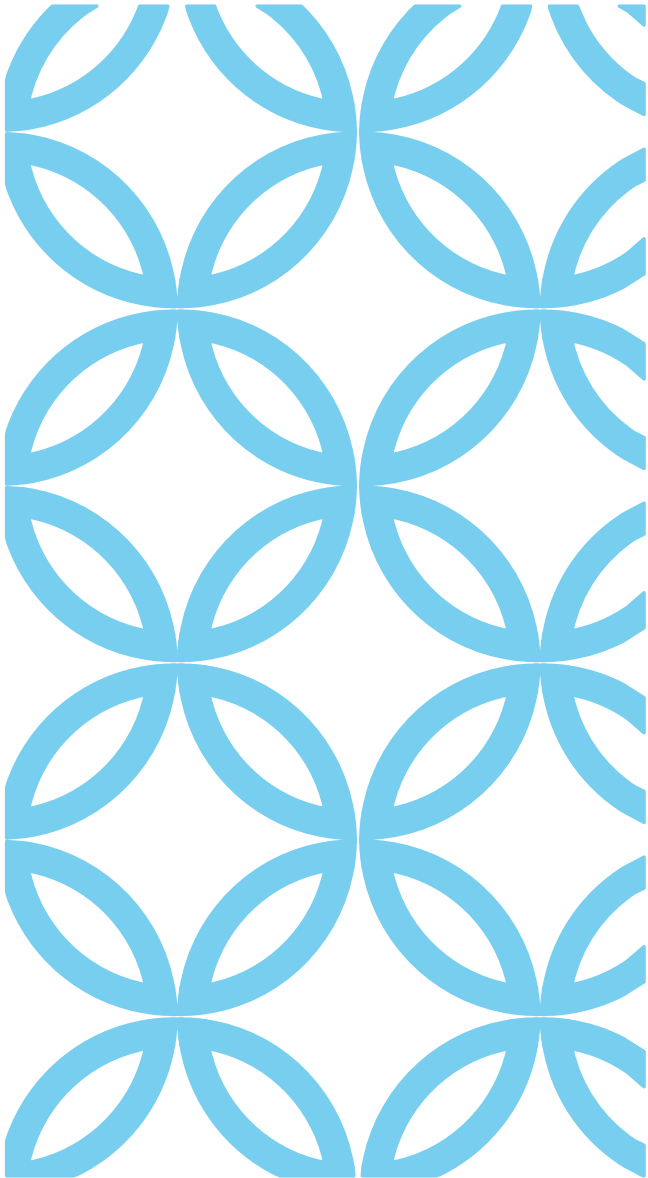
CONCEPTOS

¿ Qué es una feature engineering ?



“La ingeniería de características (feature engineering FE) Se da cuando no se ingresan los datos directamente en el algoritmo de aprendizaje automático, en su lugar, encontrará que se puede refinar partes de los datos antes de entrenar. Muchas veces el algoritmo de aprendizaje automático recompensará con un mayor rendimiento, incluso encontrará que un algoritmo simple con datos refinados generalmente supera a un algoritmo sofisticado con datos en bruto. Esta parte del flujo de trabajo de aprendizaje automático se llama ingeniería de características y es la mayor parte del tiempo una experiencia muy emocionante y gratificante”

FEATURE ENGINEERING



Propensión de compra

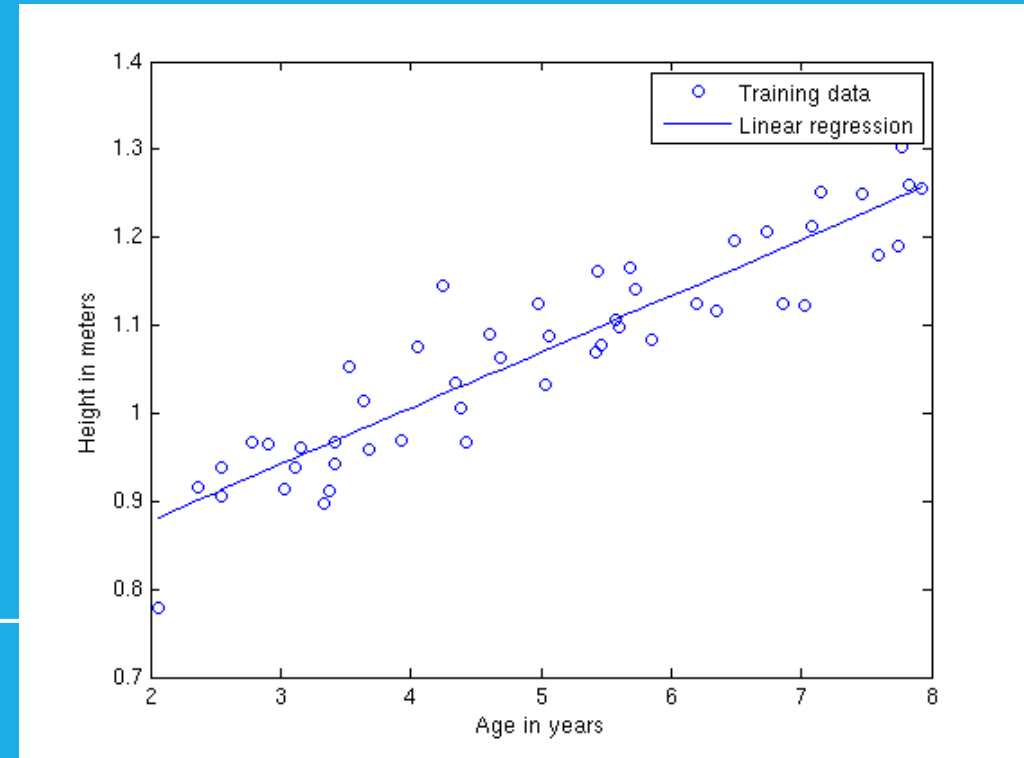
Clasificación de un tumor como benignos o malignos (Binaria)

Determinación de riesgo (alto, medio, bajo) para una solicitud de préstamo.

Sentimiento en las redes sociales como positivo, negativo o neutro

OTROS EJEMPLOS DE CLASIFICACIÓN

Objetivo: Predecir un valor numérico



REGRESIÓN

EJEMPLOS DE REGRESIÓN

- Pronóstico de ventas
- Valor de cliente a futuro
- Predecir cantidad de lluvia



Objetivo: Identificación de eventos que ocurren juntos o en secuencia



ASOCIACIÓN

DISTINCIÓN ENTRE APRENDIZAJE SUPERVISADO, NO SUPERVISADO Y REFORZADO

Aprendizaje Supervisado

- Modelos Predictivos.
- La máquina aprende explícitamente.
- Predice el futuro a partir de datos históricos.
- Resuelve problemas de clasificación y regresión.

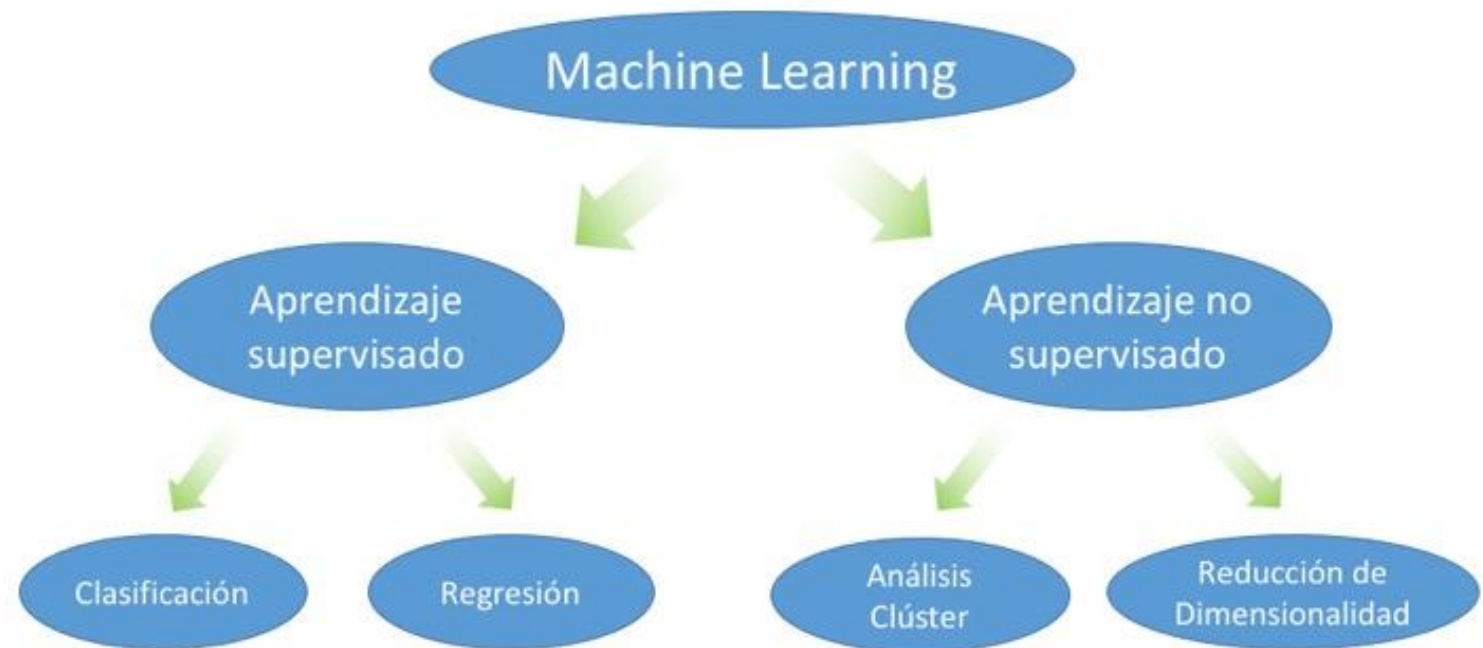
Aprendizaje No Supervisado

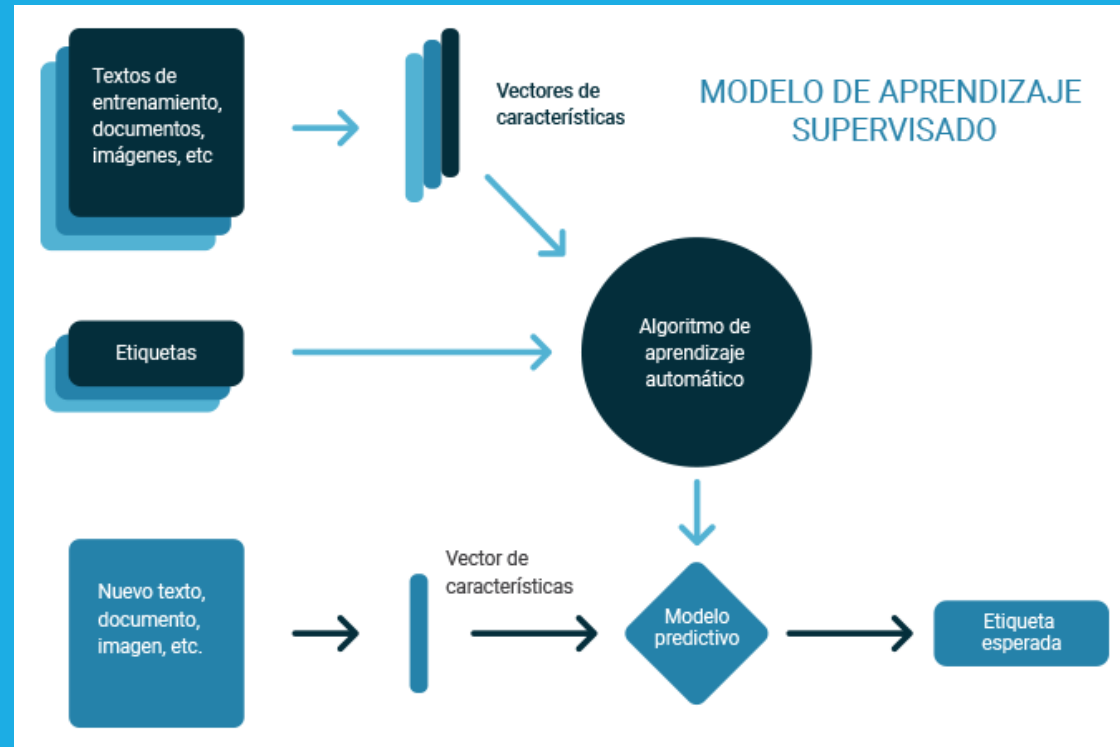
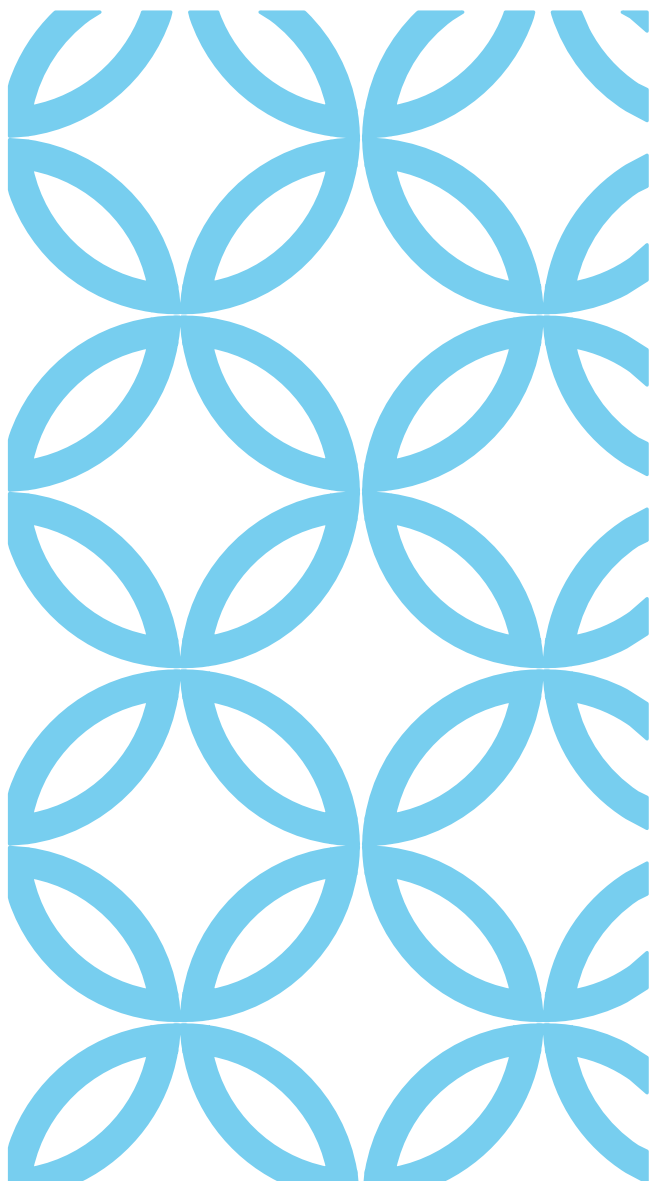
- Modelos Descriptivos.
- La máquina entiende los datos.
- La evaluación es cualitativa o indirecta.
- No realiza predicciones, encuentra algo específico.

Aprendizaje Reforzado

- Un enfoque de la IA
- Aprendizaje basado en los hallazgos.
- La máquina aprende a como actuar en un determinado entorno.
- Maximiza los hallazgos.

APRENDIZAJE SUPERVISADO Y NO SUPERVISADO

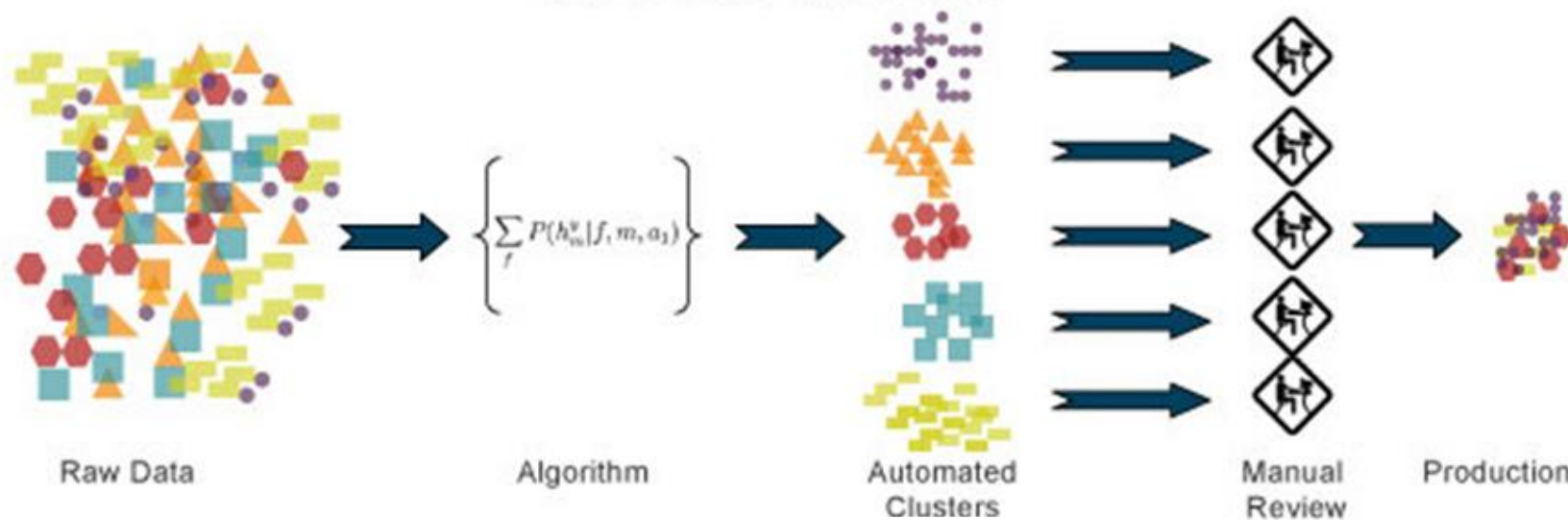




APRENDIZAJE SUPERVISADO

UNSUPERVISED LEARNING

High reliance on algorithm for raw data, large expenditure on manual review for review for relevance and coding



E-Discovery Concepts: Machine Learning

Hudson | LEGAL

APRENDIZAJE NO SUPERVISADO

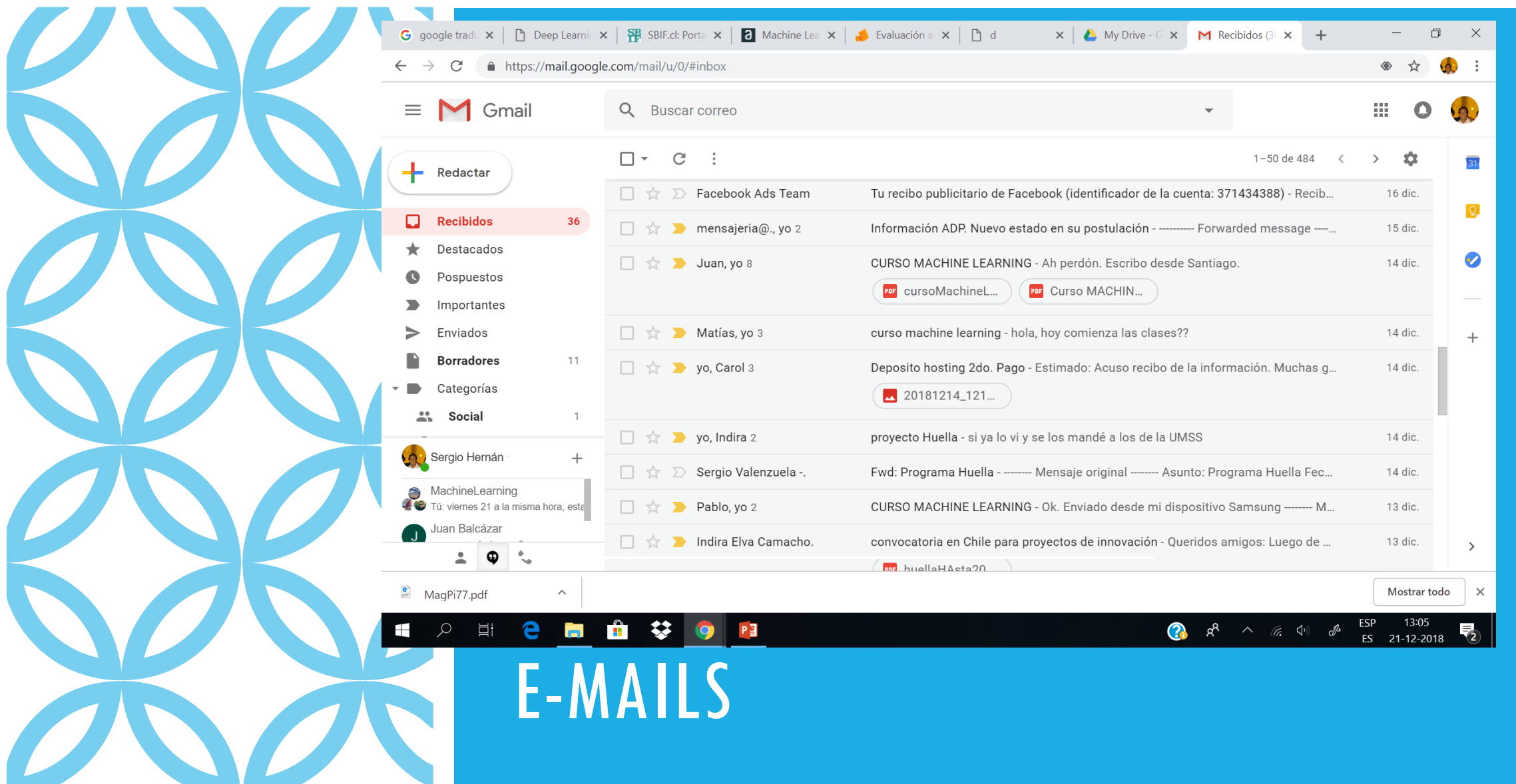
CAPÍTULO # 1

CLASIFICANDO E-MAILS, ANIMALES Y MUCHO MÁS

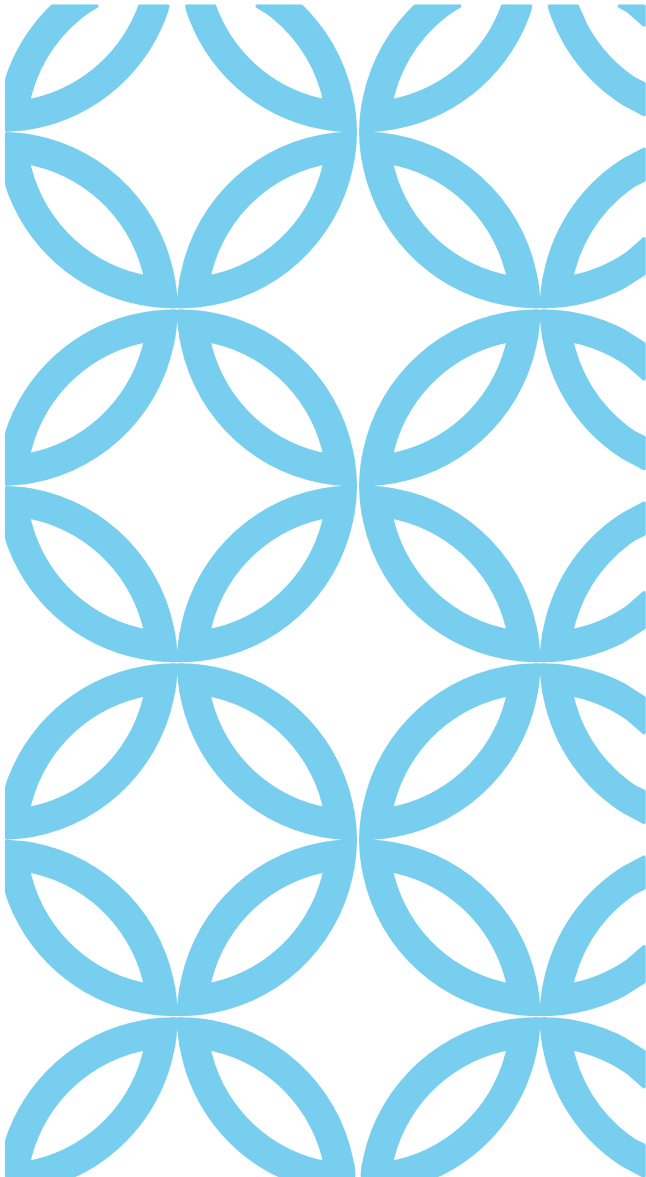
En este capítulo vamos abordar diversos problemas que podemos resolver con Machine Learning, inteligencia artificial, aprendizaje automático.

El propósito no es mostrar el formalismo, y por eso mismo no nos vamos apegar a definiciones muy formales.

El objetivo es incitar la curiosidad de utilizar algoritmos para la solución de diversos problemas.



E-MAILS



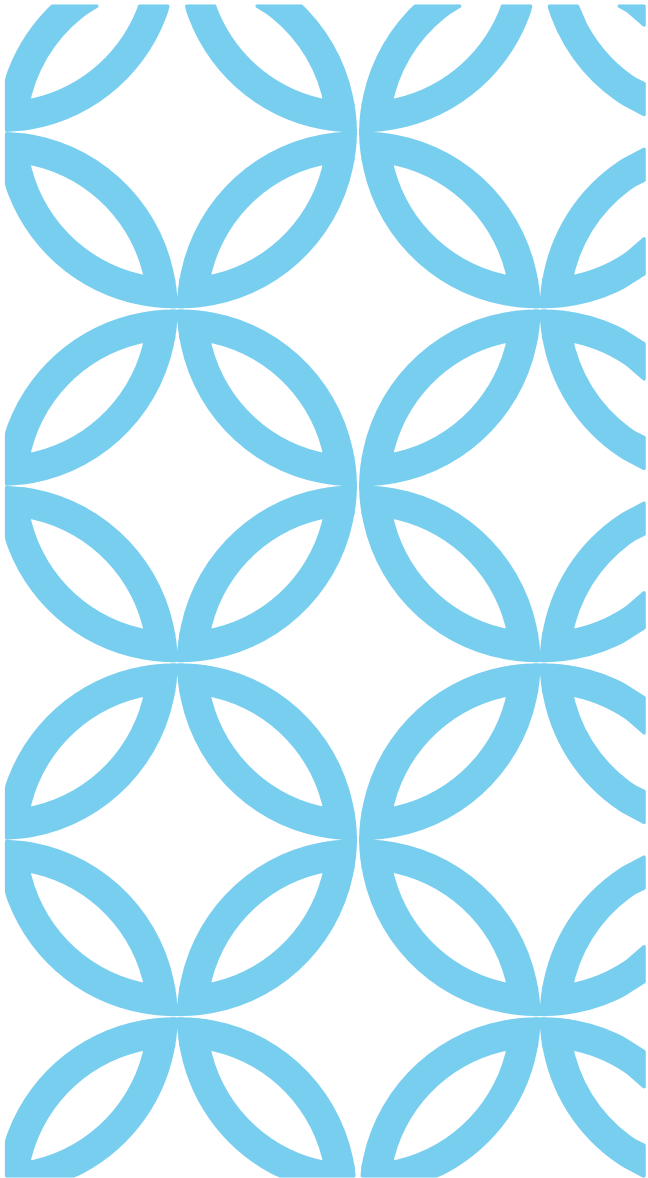
En nuestra bandeja de entrada podemos tener una gran cantidad de correos donde una parte de ellos pueden ser spam.

Alguien precisa clasificarlos y decir: “Este email dea qué es spam”.

Es necesario que alguien en internet haga eso, porque si uno lo tuviera que hacer y tener que revisar los 1.000 emails que uno recibe para decir de los 1.000 emails 990 son spam, nuestra vida se complica.

Queremos que alguien haga eso por nosotros: “Por favor, me dice si ese correo es spam o no es spam”. Ese alguien puede ser una persona que contratemos para ese propósito.

E-MAILS



Otra manera es hablarle a un programa para que haga eso por mí: “Programa por favor, clasifique este email, si él es spam o no es spam”.

Mas un programa (un computador) ... ¿sabe qué es un spam?

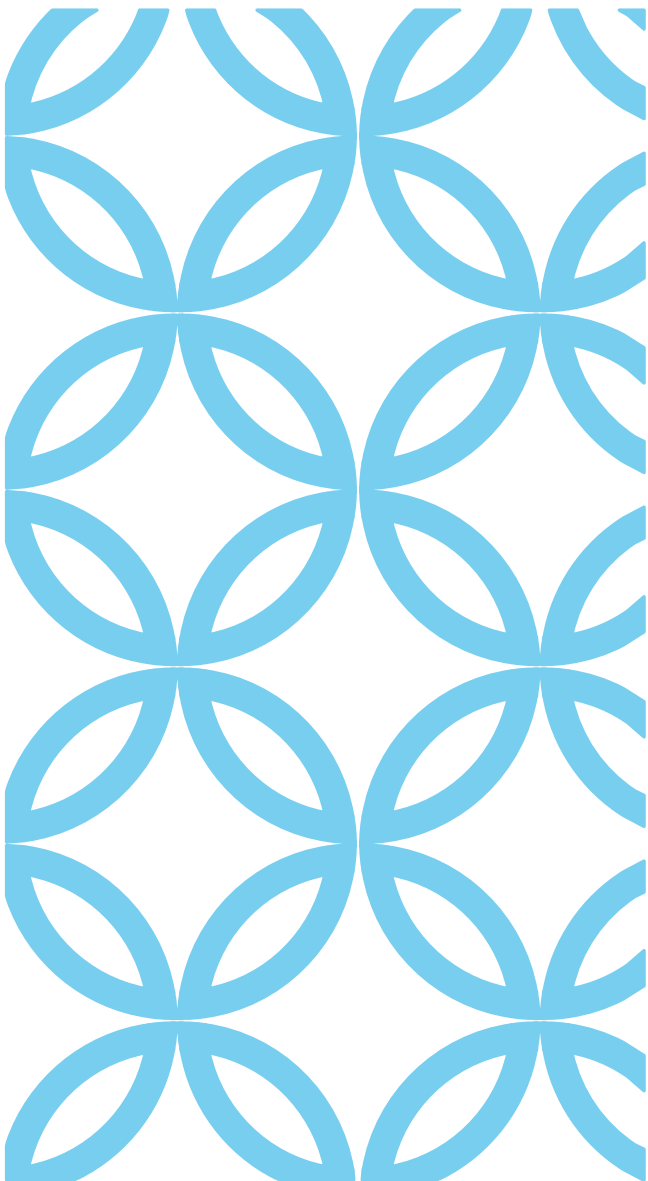
Nosotros sabemos que es un spam: “ correo no deseado que no pedimos para recibir y que alguien , el computador envió”.

Pero, el , no tiene la mínima idea de lo que significa spam,

Entonces ... ¿qué necesitamos?

.

E-MAILS



Necesitamos de alguna manera, decirle al computador que es spam y que no es spam ... más el computador no conoce la palabra spam, entonces ...

¿cómo podemos hacer eso?

¿cuál es el lenguaje que el computador conoce?

El computador entiende apenas 0 o 1, 01,2,3,4,... Binarios, numéricos, etc.

Si el computador sólo entiende esos números, entonces vamos a pedir que el computador diga si es spam es 1. Si no es spam me diga 0, y ya está resuelto el problema de la palabra spam.

Ya simplificamos para que el computador nos entienda, él ya sabe ahora !!! que si le decimos mira ese e-mail “Facturas impagas”, él es ... 1 ó 0.

Esto es, necesitamos ahora ver cada uno de los e-mails, ver y decir: “Todas impagas” (spam) es 1.

Necesitamos enseñarle al computador, ... ENTRENARLO !!!

E-MAILS

E-MAILS

¿ Cómo sabes si un email es spam o no ?

E-MAILS

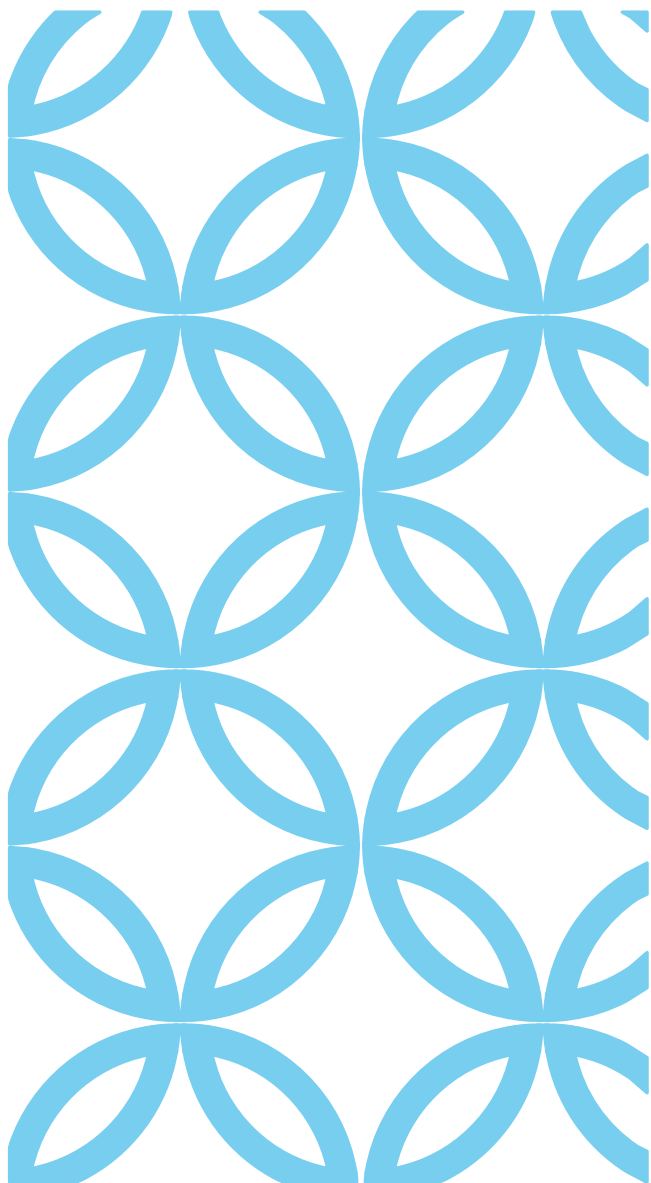
... Porque fuiste ENTRENADO.

Ya viste un millón de e-mails que eran spam y que no eran spam.

Cuando le das una mirada, ya sabes.

Ya viste tantos emails e hiciste esta CLASIFICACIÓN tanta veces, que hoy haces eso fácilmente.

Ahora el desafío es ENTRENAR al computador, para que diga: Sí es spam ... 1; si no es spam ... 0.



¿Cómo entreno al computador?

E-MAILS

E-MAILS

... Proporcionando un conjunto de emails que no son spam y otro conjunto que son spam.

Luego, le decimos “ahí está aprenda”. El computador aprende de la misma manera como aprendemos.

¿ Qué va suceder ?

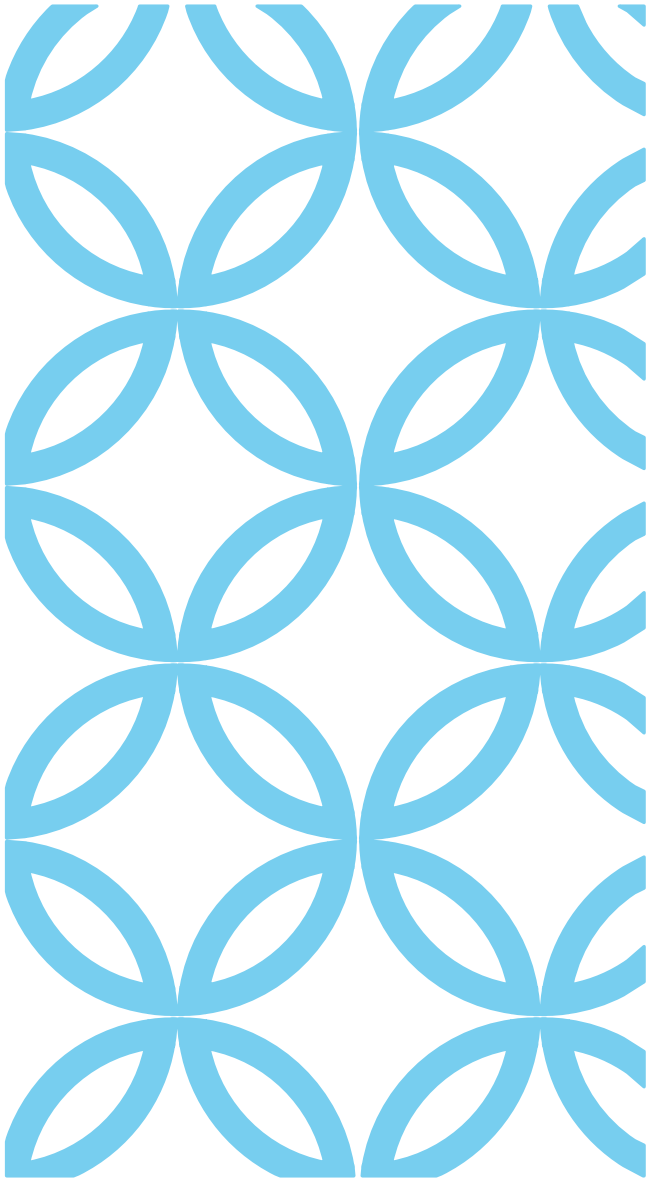
Cuando llegue un email nuevo...

¿ qué va hacer el computador ?

Él va a ver y determinar: “Es spam”. Va a ver otro email y determinar: “No es spam”.

... Y es así como funciona un programa de clasificación, él clasifica entre 0 y 1.

El computador responde cualquier pregunta de género que va a clasificar alguna cosa entre 0 y 1.



... ¿Mas que quiere decir 0?

¿O qué quiere decir 1?

En este caso en particular si es spam o no.

¿Podría querer decir otra cosa ?

... Sí, con mucha certeza.

E-MAILS

CLASIFICACIÓN

Podemos usar la CLASIFICACIÓN entre 0 y 1 para diversas cosas y también en diversas situaciones.

Vamos a citar otros ejemplos:

¿Cómo implemento un algoritmo si no entiendo yo como hago una cosa?.

“Quiero implementar un algoritmo de suma”.
Ok, ¿cómo hacemos la suma?.

Ahora quiero implementar un algoritmo que diga si es spam o no es spam, o si un cliente va a comprar o no, si un empleado va a renunciar o no, entonces ... necesito saber cómo hago eso !!! Esto es como yo, Sergio Hernán, clasifico eso.

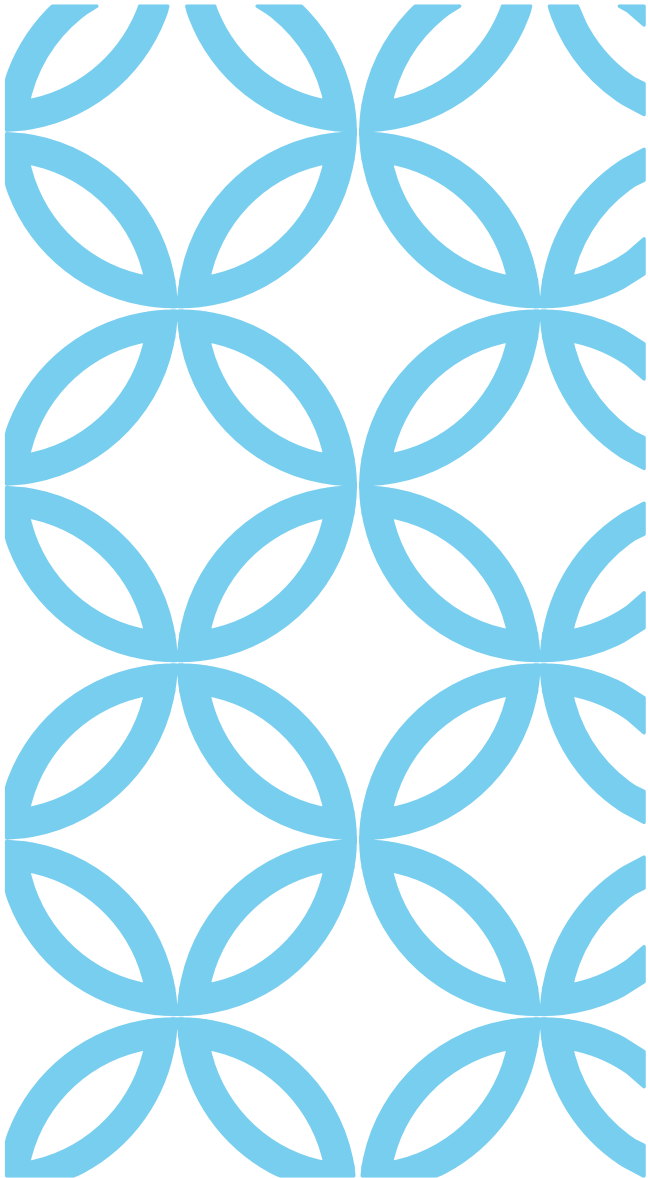
CLASIFICACIÓN

Vamos a tomar un ejemplo de la vida cotidiana.

Queremos clasificar entre 0 y 1.

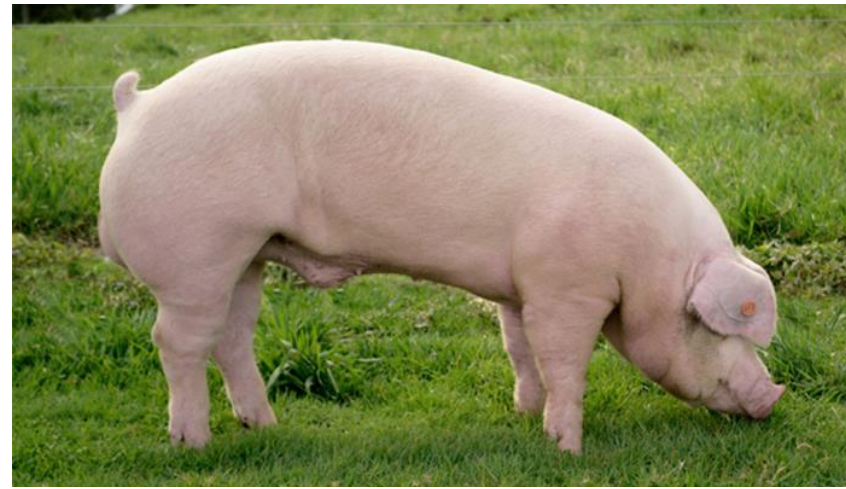
Cuando era niño e iba al campo, a veces encontraba un animal como este:





¿ Qué animal es este ?
CLASIFICACIÓN

CLASIFICACIÓN



...Bien acabamos de clasificar y lo clasificamos como un ... cerdo.

¿Por qué?

Noten la pierna de él, es una pierna corta.

¿Qué más?

Él es medio gordo, cerdito es gordito y dice “oink, oink”, entonces Podemos concluir que es un cerdo.

CLASIFICACIÓN

...Más también en el campo como es de costumbre, existen diversos tipos de animales y a veces me encontraba un animal como éste:



¿ Qué animal es éste ?

CLASIFICACIÓN



... En el momento que lo observaba yo me preguntaba y decía:

“Que chanchito tan bonito y gordito” ... No !!!

Yo decía que perrito tan simpático”.

¿ Por qué ?

Pues, en mi cabeza ocurrió el proceso de CLASIFICACIÓN.

CLASIFICACIÓN



... Cuando observaba el perro yo pensaba:
¿Será que tiene pierna corta igual que el cerdo ? ... No.

¿Será que él es gordo igual que el chanchito?
... Algunos sí otros no.

¿Será que él dice oink oink igual que el chanchito? ... Él no decía oink, oink.

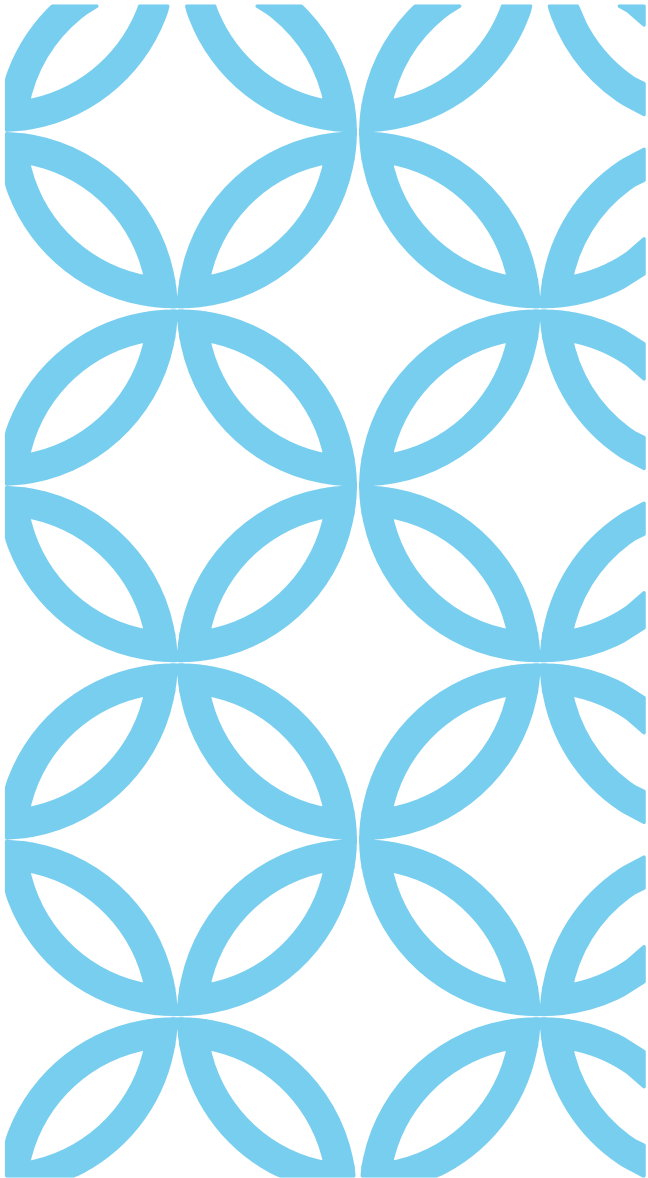
Entonces él no es un cerdo; él es un perro.

CLASIFICACIÓN

De acuerdo con las tres características que citamos:







- Pierna corta
- Es gordo
- Dice oink, oink

Podemos clasificar entre un cerdo y un perro.



¿ Cómo le
enseñamos al
computador hacer
la clasificación ?

EJEMPLO: ANIMALES

animal	¿ pierna corta ?	¿ gordo ?	¿ dice oink oink?	¿ qué es ?
				
				
				
				
				
				

CLASIFICANDO ANIMALES

A partir de la tabla anterior con fotos de varios animales, queremos saber:

- si tienen pierna corta o no
- si son gordos o no
- si dicen oink oink o no
- también conocer si son cerdos o perros.

Usaremos 1 para decir si es un cerdo y 0 si es perro.

¿ vamos a llenar la tabla ?

Veamos el primer animal:



**FIGURA 1.1:
PRIMER ANIMAL**

¿ Él tiene pierna corta ? ... Sí, anotamos 1.

¿ Él es gordo ? ... Sí, asignamos 1.

¿ Él dice oink oink ? ... Sí, marcamos 1.

Así la línea queda:

[1, 1, 1, 1]

Veamos el próximo animal:



**FIGURA 1.2:
SEGUNDO ANIMAL**

¿ Él tiene pierna corta ? ... Sí, anotamos 1.

¿ Él es gordo ? ... No, asignamos 0.

¿ Él dice oink oink ? ... Sí, marcamos 1.

Así la línea queda:

[1, 0, 1, 1]

Veamos el próximo animal:



**FIGURA 1.3:
TERCER ANIMAL**

¿ Él tiene pierna corta ? ... Sí, anotamos 1.

¿ Él es gordo ? ... Sí, asignamos 1.

¿ Él dice oink oink ? ... No, marcamos 0.

Así la línea queda:

[1, 1, 0, 1]

Veamos como está nuestra tabla:

animal	¿ pierna corta ?	¿ gordo ?	¿ dice oink oink?	¿ qué es ?
	1	1	1	1
	1	0	1	1
	1	1	0	1

Para cada elemento de la tabla,
anotamos la característica (feature), osea,
si tiene pierna corta, si es gordo o si dice oink oink, y
diciendo apenas **sí** o **no** anotamos 1 ó 0
¿ y al final qué tipo de animal es ?
... si es cerdo (1) o perro (0).
Todavía faltan tres animales que debemos analizar ...
¿ continuamos ?



**FIGURA 1.4:
CUARTO ANIMAL**

¿ Él tiene pierna corta ? ... No, anotamos 0.

¿ Él es gordo ? ... No, asignamos 0.

¿ Él dice oink oink ? ... No, marcamos 0.

Así la línea queda:

[0, 0, 0, 0]

Veamos el próximo animal:



**FIGURA 1.5:
QUINTO ANIMAL**

¿ Él tiene pierna corta ? ... Sí, anotamos 1.

¿ Él es gordo ? ... Sí, asignamos 1.

¿ Él dice oink oink ? ... Sí, marcamos 1.

Así la línea queda:

[1, 1, 1, 0]

Veamos el próximo animal:



**FIGURA 1.6:
SEXTO ANIMAL**

¿ Él tiene pierna corta ? ... No, anotamos 0.




¿ Él es gordo ? ... no, asignamos 0.

¿ Él dice oink oink ? ... No, marcamos 0.

Así la línea queda:

[0, 0, 0, 0]

Veamos ahora como queda la tabla:

animal	¿pierna corta?	¿ gordo ?	¿dice oink oink?	¿ qué es ?
	1	1	1	1
	1	0	1	1
	1	1	0	1
	0	0	0	0
	1	1	1	0
	0	0	0	0

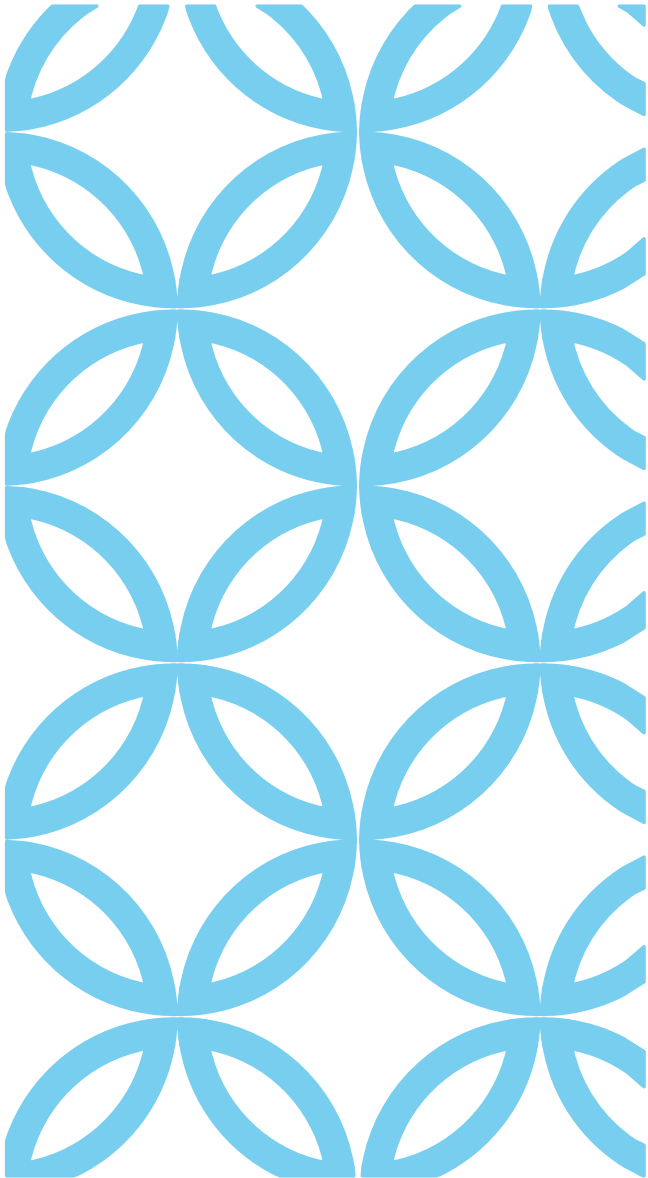
CLASIFICACIÓN ANIMALES Y EMAILS

La anterior tabla define las características de los animales por medio de las tres primeras columnas (pierna corta, gordo, dice oink oink).

La última columna (¿ qué es ?) clasifica el animal.

Estamos usando las características (features) para clasificar un animal entre dos tipos.

Podríamos usar esas características u otras para clasificar un e-mail entre spam y no spam, por ejemplo: tamaño e-mail; palabras que aparecen; horario en que fue enviado; si conozco o no al remitente; si es la primera vez que llega ese e-mail.



En el cotidiano día a día en el campo usamos las tres características descritas anteriormente para clasificar animales, pero; podríamos usar otras como:

¿ es color rosa ?

¿ es color negro ?

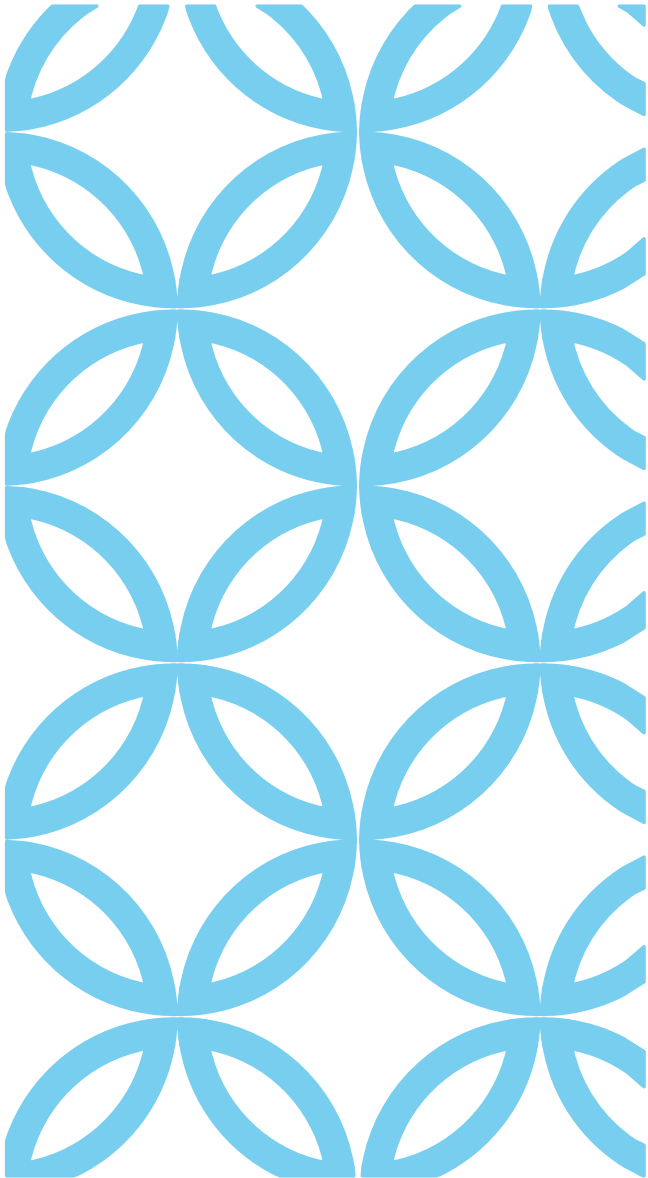
¿ es color blanco ?

¿ es color azul ?

¿ es color X ?

El color puede influir para para decir si es un animal u otro. Existen animales de un color y también de otro color.

CLASIFICACIÓN



Existen diversas características que podemos usar para clasificar un animal.

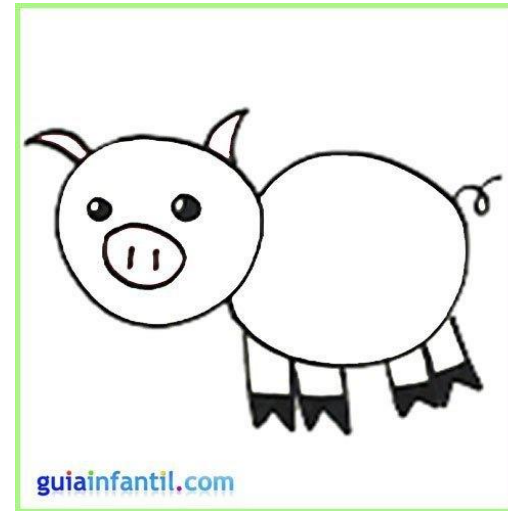
Repare que todos los problemas de CLASIFICACIÓN hacen eso: dado un conjunto de características y cúmulo de datos, podemos entrenar nuestro algoritmo, nuestro programa.

CLASIFICACIÓN

ENTRENAMIENTO

Dada la tabla de características que utilizamos para entrenar nuestro modelo.

Ahora podemos preguntar a nuestro modelo.

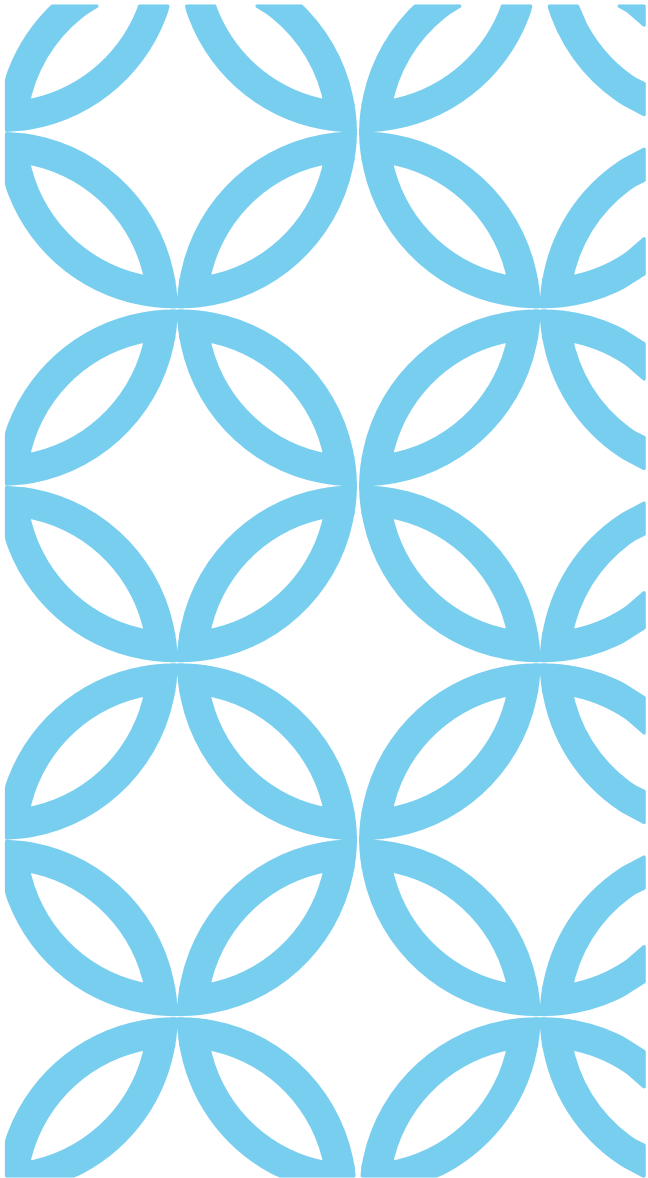


¿ y éste de aquí , quién es ?



¿ En base a las características definidas es un perro (0) o un cerdo (1) ?

Programa toma una decisión basada en toda la experiencia anterior que tuvo
... 1



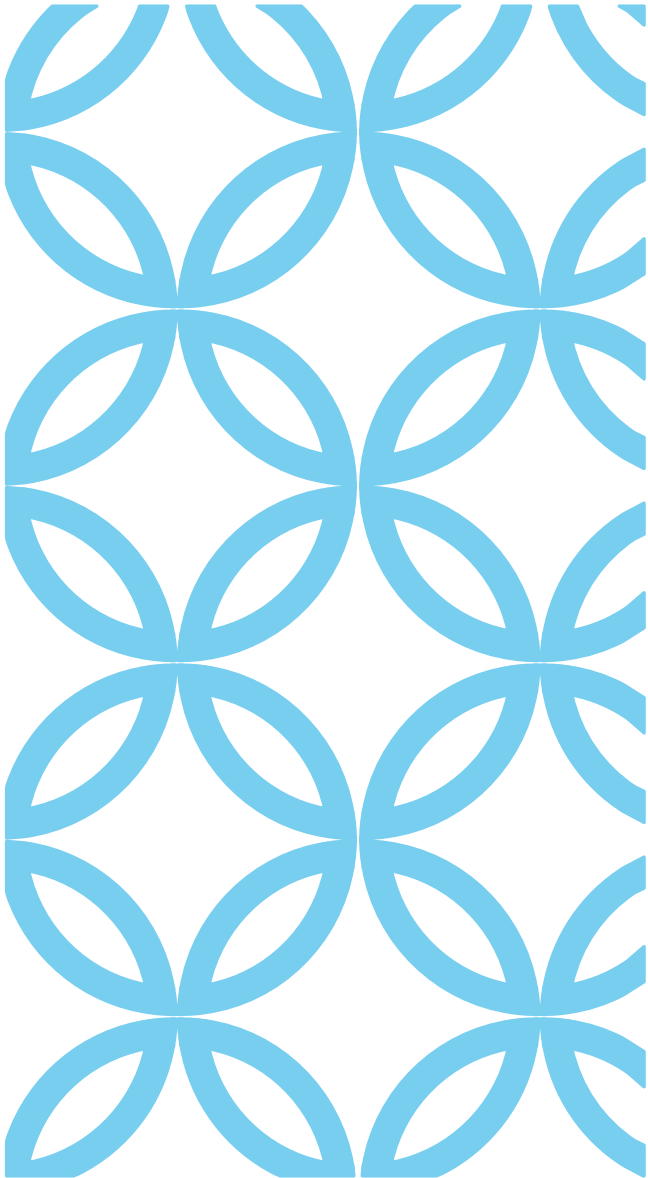
Los algoritmos están basados en características y usaremos entre 0 y 1 para poder definirlas.

Es claro que veremos otros tipos después, más por ahora para CLASIFICAR si es o no un animal que esperamos, y es exactamente eso lo que haremos con los algoritmos.

RESUMIENDO ...

... AHORA EMPIEZA LO BUENO !!!





Queremos implementar un sistema de CLASIFICACIÓN que consiga clasificar alguna cosa entre:

0 y 1

-1 y 1

1 y 2

A y B

Entre dos categorías diferentes: Cerdo y Perro.

VAMOS PARA EL CÓDIGO ...



PYTHON

Vamos a comenzar a instalar algunas bibliotecas en Python con **pip3** (instalador de paquetes en Python).

La biblioteca que usaremos es:

Scikit-learn

Aprovecharemos para instalar dos dependencias que scikit usará internamente:

> **sudo apt-get install python3-sklearn**

En caso de usar Linux, será necesario anteponer el comando **sudo**.



PYTHON

Además de la biblioteca de Python scikit-learn, también tenemos acceso a la documentación:

<http://scikit-learn.org/stable/documentation.html>

Ahora vamos clasificar entre diversos cerdos y perros. Abrimos un editor de texto (utilice el de su preferencia) para escribir código Python.

Creamos un archivo llamado:

`clasificacion.py`

Y lo guardamos. Asegúrese de guardar el archivo dentro de un directorio de fácil acceso a la terminal.

En este archivo vamos a definir todos los cerdos y perros que ya conocemos.



PYTHON

Por ejemplo, queremos representar nuestro primer cerdo, entonces escribimos:

`cerdo1`

¿ y cómo era ese cerdo ?

Él era un array de 3 valores,

¿ recuerdan ?

¿ cuáles son los valores ?

0 ó 1, él tiene o no una característica, más ... ¿ y cuáles son las tres características que yo Sergio Hernán decidí usar ahora ?

1. Si tiene pierna corta
2. Si es gordo
3. Si dice guau guau



PYTHON

¿ tiene pierna corta ? ... Sí.

```
cerdo1 = [ 1 ]
```

¿ es gordo ? ... Sí.

```
cerdo1 = [ 1, 1 ]
```

¿ dice guau guau ? ... No.

```
cerdo1 = [ 1, 1, 0 ]
```



PYTHON

Ahora vamos a colocar el segundo y tercer cerdo.

Ellos también tienen pierna corta, son gordos y dicen guau guau.

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```



PYTHON

Ya registramos los tres cerdos, ahora necesitamos registrar los tres perros.

El primer perro tiene pierna corta, es gordo y no dice guau guau.

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```



PYTHON

Los otros dos perros tienen pierna corta, no son gordos y no dicen guau guau.

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

```
perro3 = [ 1, 0, 1 ]
```



PYTHON

Ya sabemos que del primero al tercero son cerdos y del cuarto al sexto son perros. Ahora tenemos que decir si son 1 ó 0; si son 1 ó -1. Necesitamos de todos esos elementos y los llamaremos de **datos**:

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

```
perro3 = [ 1, 0, 1 ]
```

```
datos=[cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
```



PYTHON

Ahora que tenemos agrupados nuestros datos en un array, necesitamos etiquetar para indicar que es cada uno de esos elementos. Usaremos la variable **etiquetas** para indicar lo que representa cada uno.

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

```
perro3 = [ 1, 0, 1 ]
```

```
datos=[cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
```

```
etiquetas=[1,1,1,-1,-1,-1]
```



PYTHON

Para etiquetar como perro, podemos asignar otro número. Puedo marcar con 0, -1, 10.000 o con lo que yo quiera. Para enfatizar en la distinción de los dos elementos, o sea decir que uno es opuesto del otro usaremos 1 positivo (1) para cerdo | 1 negativo (-1) para perro.

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

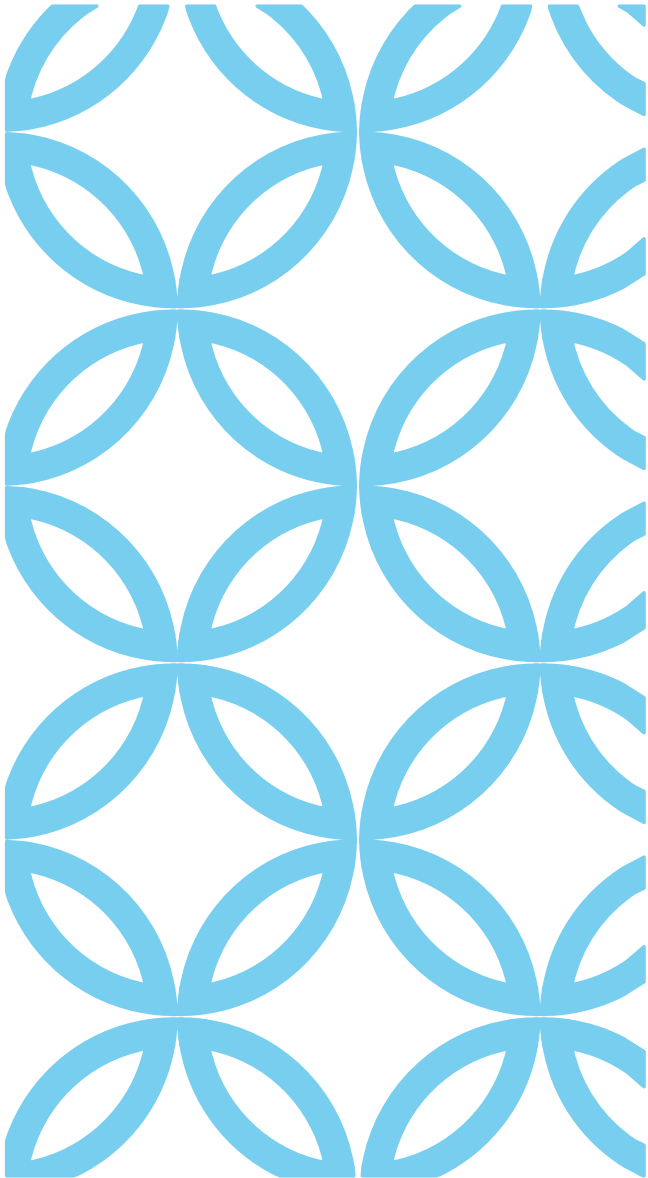
```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

```
perro3 = [ 1, 0, 1 ]
```

```
datos=[cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
```

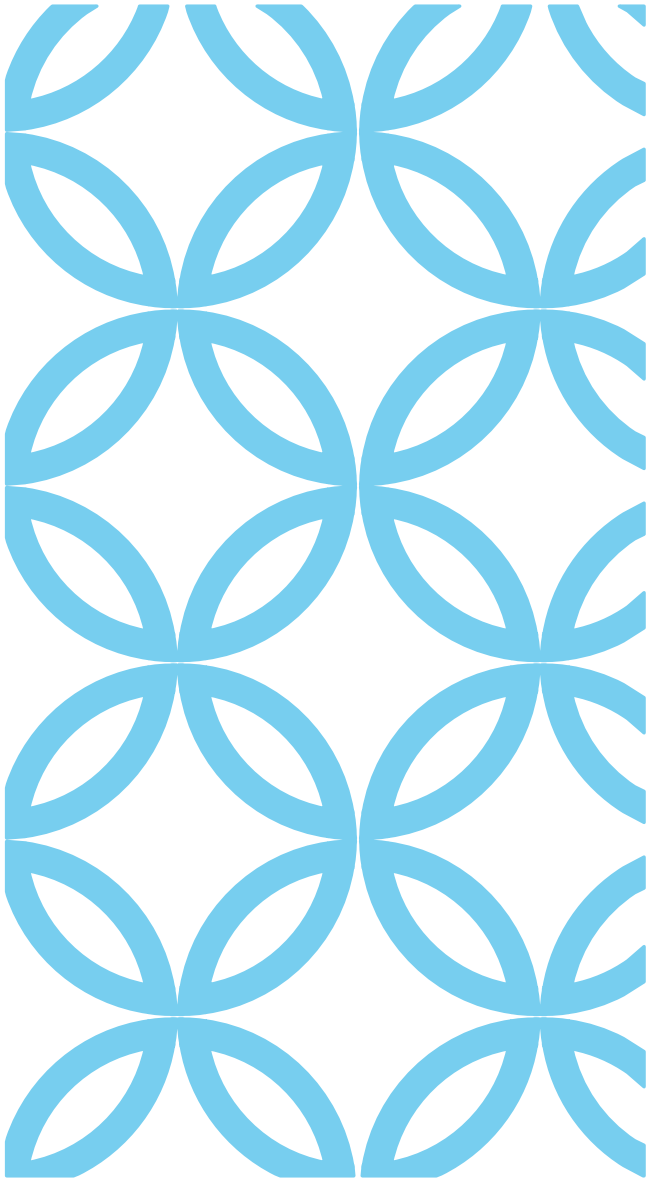
```
etiquetas=[1,1,1,-1,-1,-1]
```

Ya marcamos todos nuestros elementos,
pero Tenemos ahora un elemento
misterioso: un animal que es gordo, tiene
pierna corta y dice guau-guau.

misterioso = [1, 1, 1]

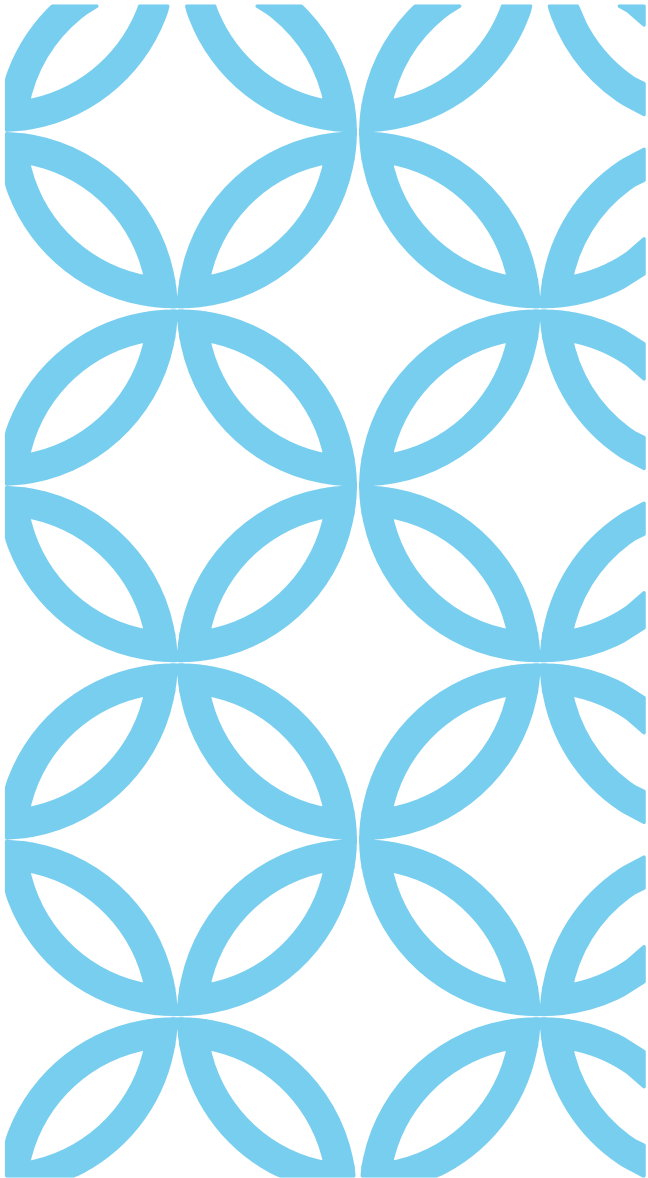
SKLEARN



... ¿ y ahora ?

¿ este animal es un cerdo o un
perro ?

SKLEARN

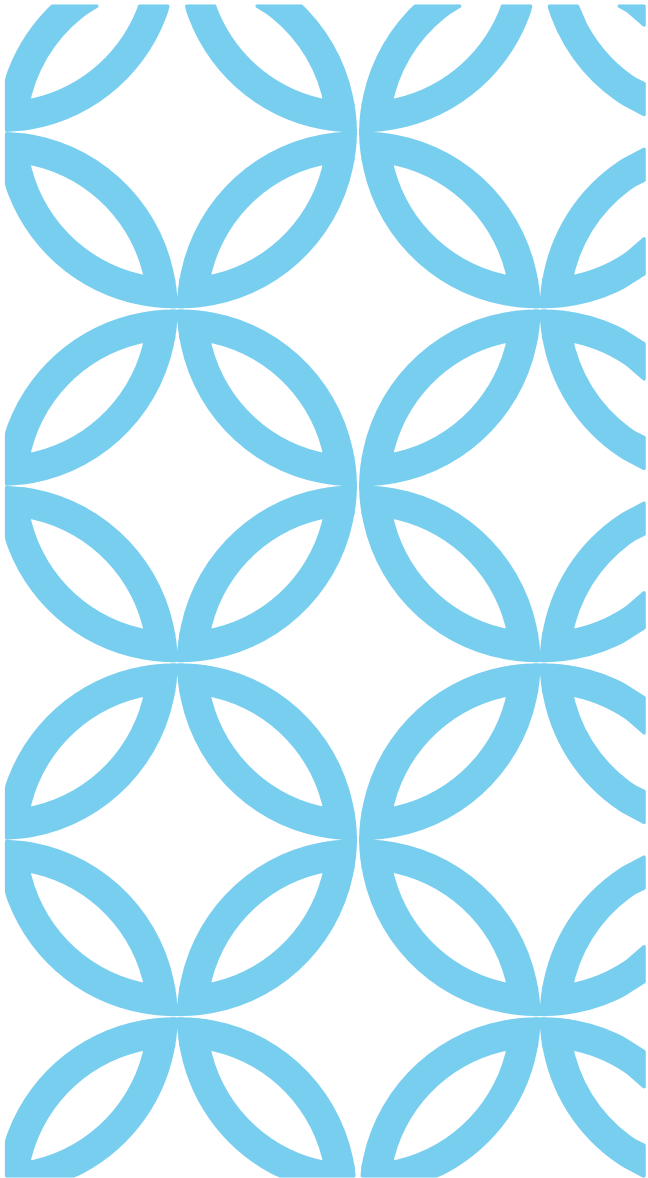


Lo que queremos saber es:

¿ será que él es un perro o un cerdo ?

Entonces queremos entrenar un algoritmo de clasificación basado en esos datos para que nos diga si es un perro o un cerdo.

SKLEARN



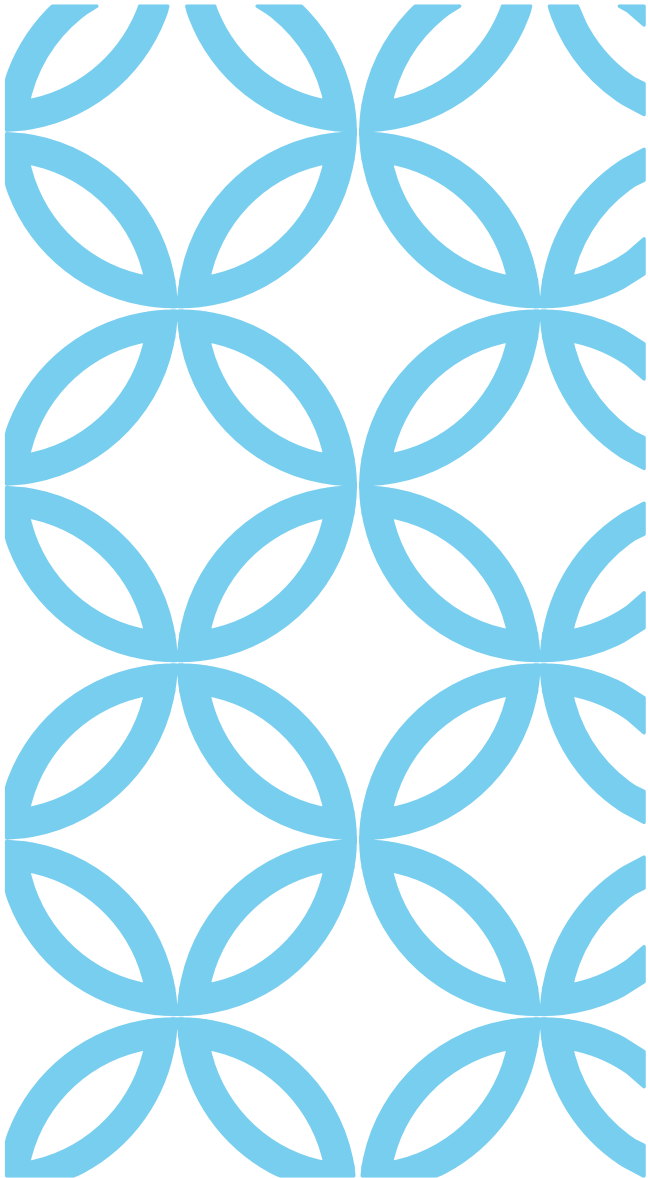
En Python la biblioteca mas famosa que hace ese trabajo es **sklearn**.

Esta biblioteca hace el entrenamiento basado en en el algoritmo bayesiano, llamado naive_bayes.

Vamos a importar el algoritmo MultinomialNB, NB de naive bayes:

```
from sklearn.naive_bayes import MultinomialNB
```

SKLEARN



Multinomial es el algoritmo que utilizaremos para entrenar nuestro modelo que dirá si nuestros elementos son perros o cerdos.

Para entrenar este modelo, necesitaremos de nuestros datos y etiquetas.

¿ ... y cómo hacemos para crear un modelo ?

Basta apenas hacer una llamada para el MultinomialNB:

SKLEARN



PYTHON

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

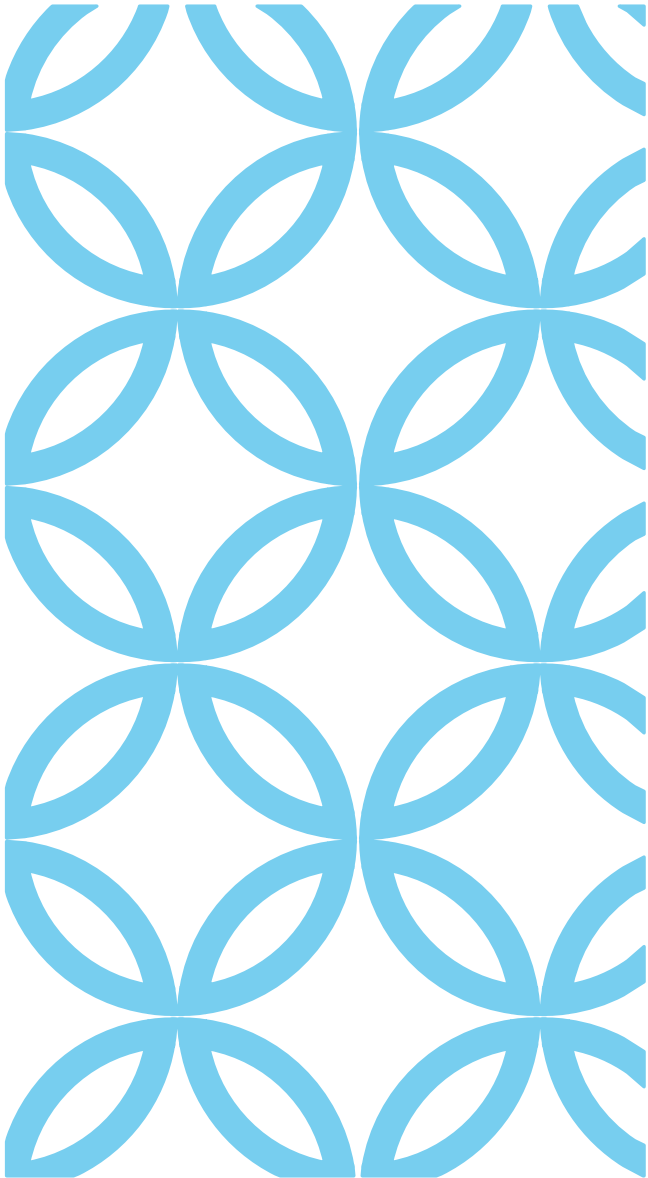
```
perro3 = [ 1, 0, 1 ]
```

```
datos = [cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
```

```
etiquetas = [1,1,1,-1,-1,-1]
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
modelo = MultinomialNB()
```



Creamos un modelo, vamos a correr nuestro código.
Guarda el script anterior de la anterior diapositiva con el nombre de:

`clasificacion.py`

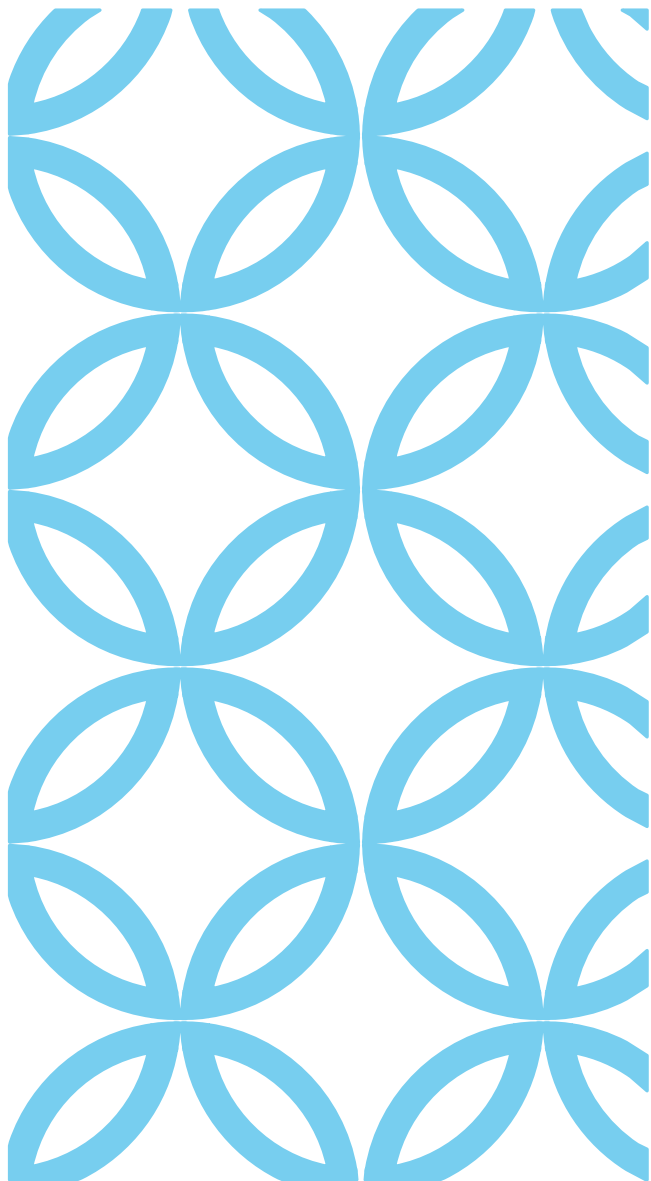
Ahora vamos a ejecutar el archivo:

```
pi@raspberrypiML2:~ $ cd ML
```

```
pi@raspberrypiML2:~/ML $ python3 clasificacion.py
```

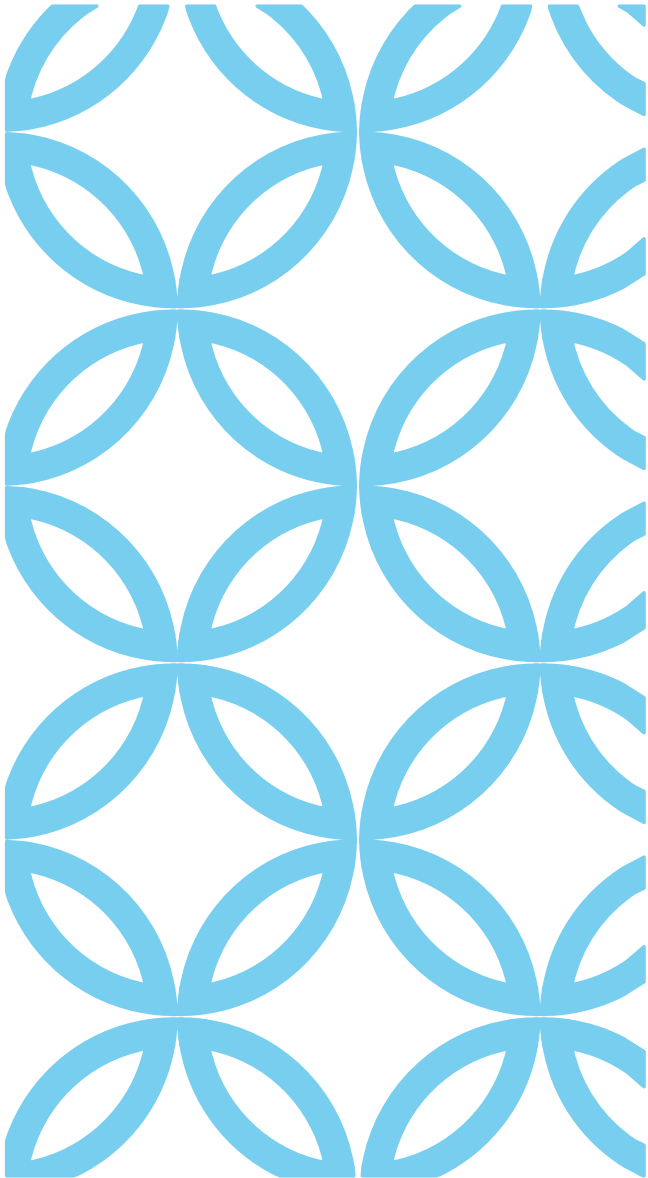
```
pi@raspberrypiML2:~/ML $
```

SKLEARN



¿ sucedió algo ?

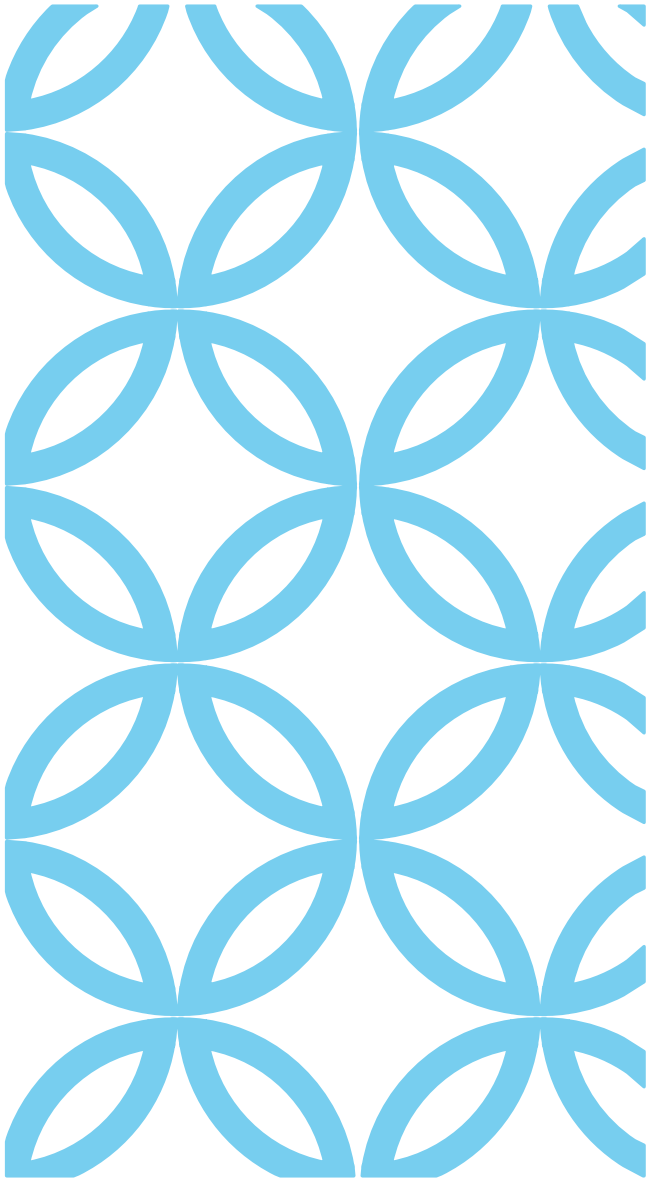
SKLEARN



Claro que no !!!

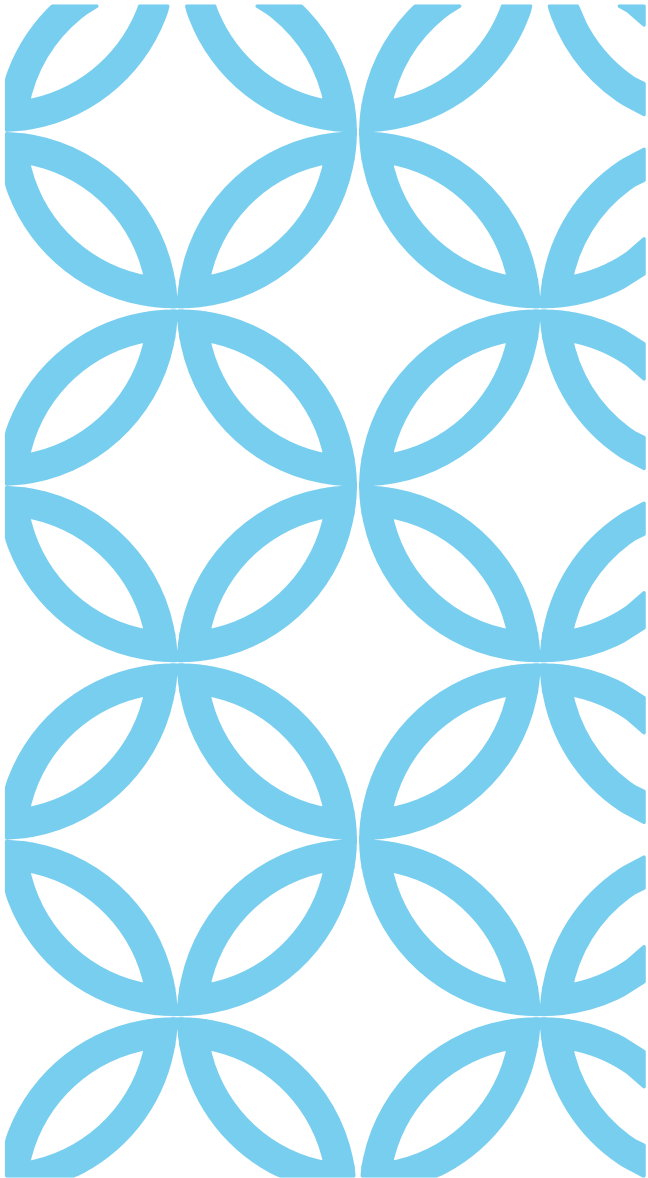
Apenas creamos el modelo, todavía
no lo entrenamos.

SKLEARN



¿ Cómo nosotros los humanos
entrenamos el aprendizaje ?

SKLEARN

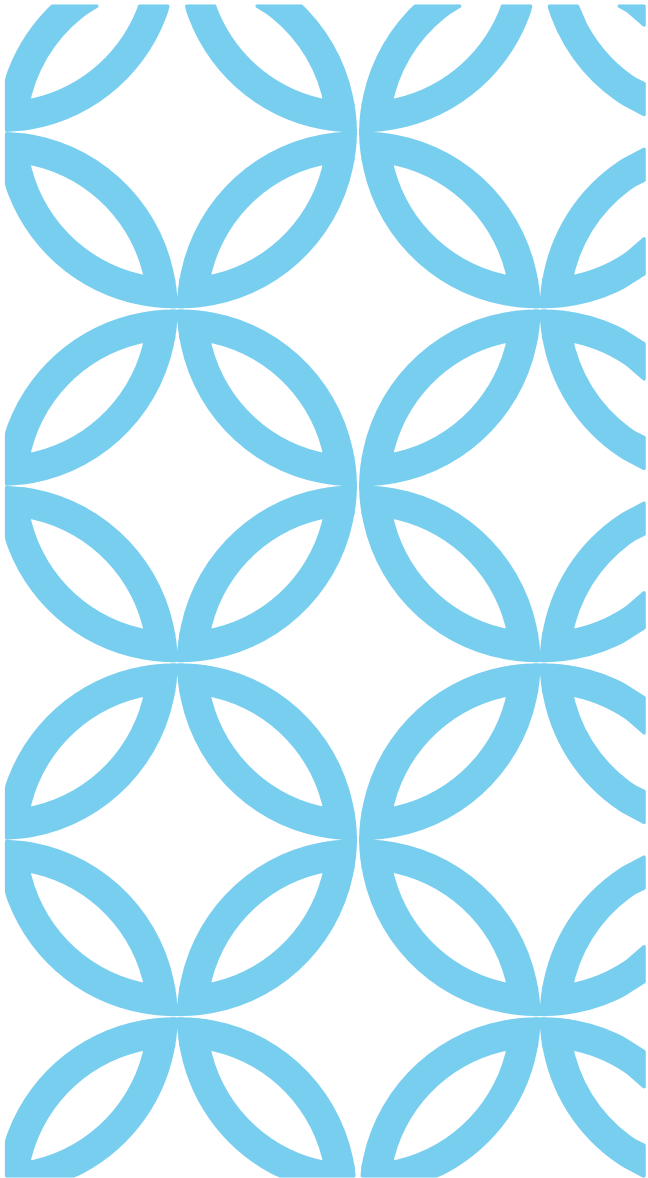


Vemos un elemento y clasificamos como cerdo, observamos otro y clasificamos como perro, esto es basado en los datos y en las etiquetas, nosotros entrenamos el modelo de nuestro cerebro, y necesitamos hacer la misma cosa para nuestro modelo.

```
modelo = MultinomialNB()
```

```
modelo.entrenar (datos, etiquetas)
```

SKLEARN



Pero !!! ...

No existe el método entrenar,
¿ entonces cómo lo hacemos ?

Observen que nuestro modelo necesita adecuarse a los datos y etiquetas, o sea los datos de cerdo1 al cerdo3 son 1; los datos de perro4 al perro6 son -1. Y de esta maenra precisamos **adecuar** nuestro modelo, para ello utilizaremos el método adecuar, en inglés el método **fit**:

`modelo.fit (datos, etiquetas)`

SKLEARN



PYTHON

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

```
perro3 = [ 1, 0, 1 ]
```

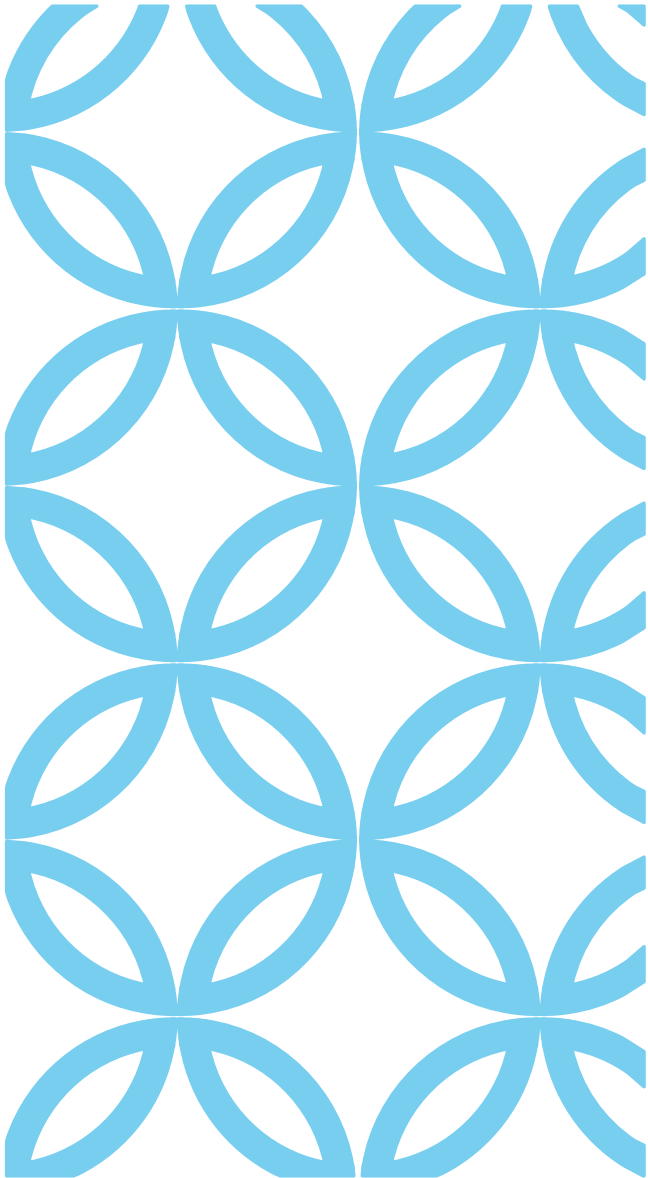
```
datos = [cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
```

```
etiquetas = [1,1,1,-1,-1,-1]
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
modelo = MultinomialNB()
```

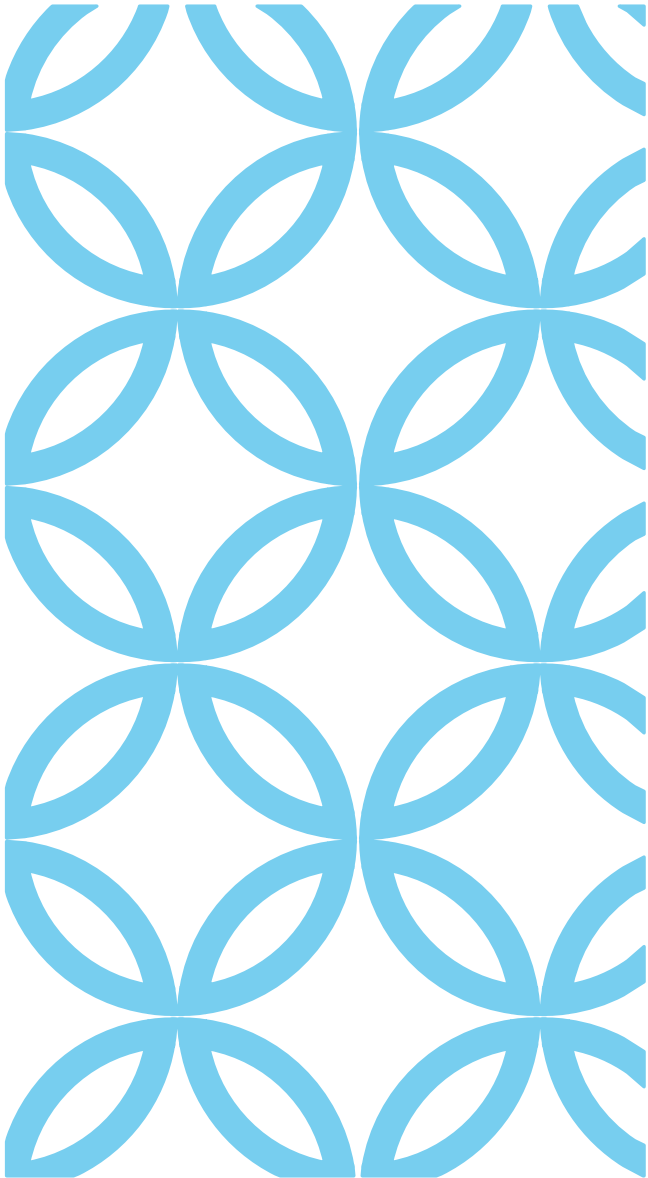
```
modelo.fit(datos, etiquetas)
```



¿ vamos a correr de nuevo el código ?

> python3 clasificacion.py

SKLEARN



¿ sucedió algo ?

No pasó nada, ya que sólo entrenamos el modelo y lo que necesitamos es pedir que el modelo **prediga** quién es el elemento misterioso, utilizando el método *predict*:

```
modelo.predict(misterioso)
```

Este método devuelve si es un perro o un cerdo, entonces necesitamos imprimir con la sentencia *print*:

SKLEARN



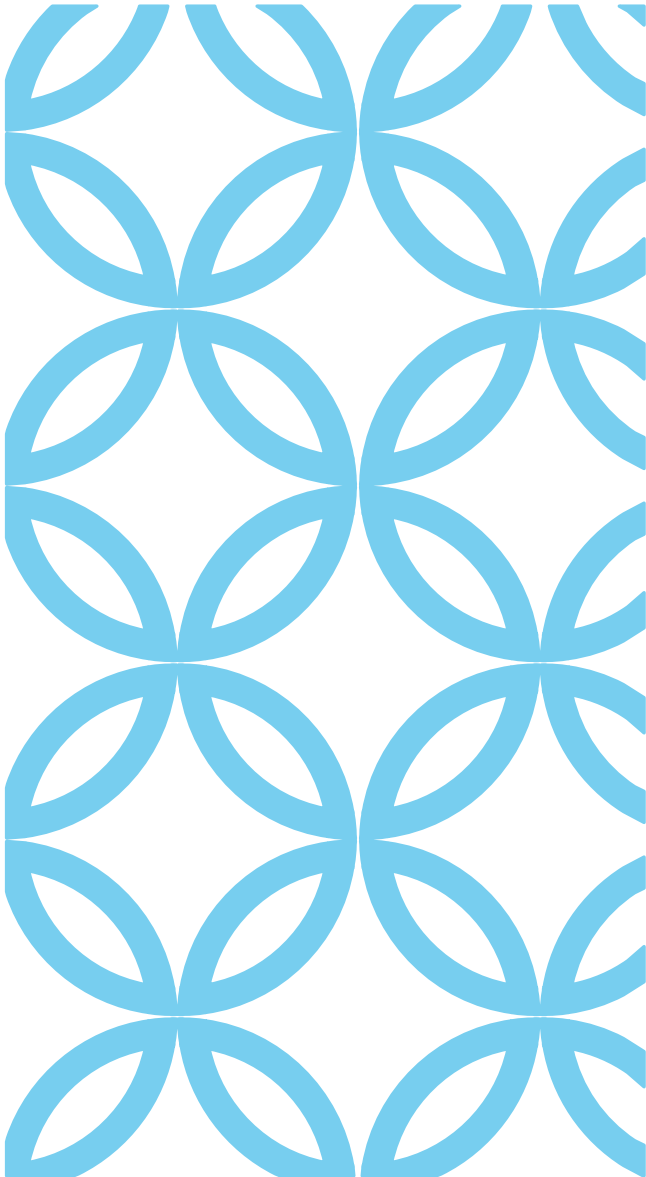
PYTHON

```
# [ es gordito ? Tiene pierna corta? Dice guau guau ?]
cerdo1 = [ 1, 1, 0 ]
cerdo2 = [ 1, 1, 0 ]
cerdo3 = [ 1, 1, 0 ]
perro1 = [ 1, 1, 1 ]
perro2 = [ 1, 0, 1 ]
perro3 = [ 1, 0, 1 ]

datos = [cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
etiquetas = [1,1,1,-1,-1,-1]

misterioso = [ 1, 1, 1 ]

from sklearn.naive_bayes import MultinomialNB
modelo = MultinomialNB()
modelo.fit(datos, etiquetas)
print(modelo.predict(misterioso))
```

probando nuevamente nuestro algoritmo:

```
pi@raspberrypiML2:~/ML $ python3 clasificacion.py
```

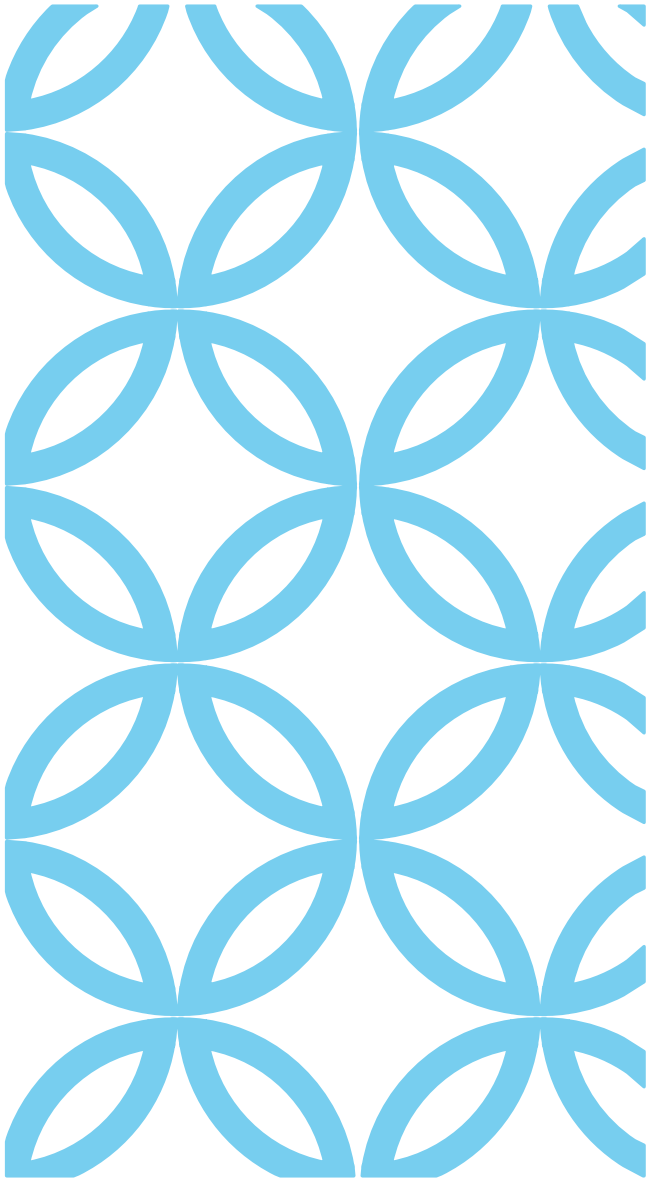
```
/usr/lib/python3/dist-packages/sklearn/utils/validation.py:395:  
DeprecationWarning: Passing 1d arrays as data is deprecated in  
0.17 and will raise ValueError in 0.19. Reshape your data either  
using X.reshape(-1, 1) if your data has a single feature or  
X.reshape(1, -1) if it contains a single sample.
```

```
DeprecationWarning)
```

```
[-1]
```

```
pi@raspberrypiML2:~/ML $
```

SKLEARN



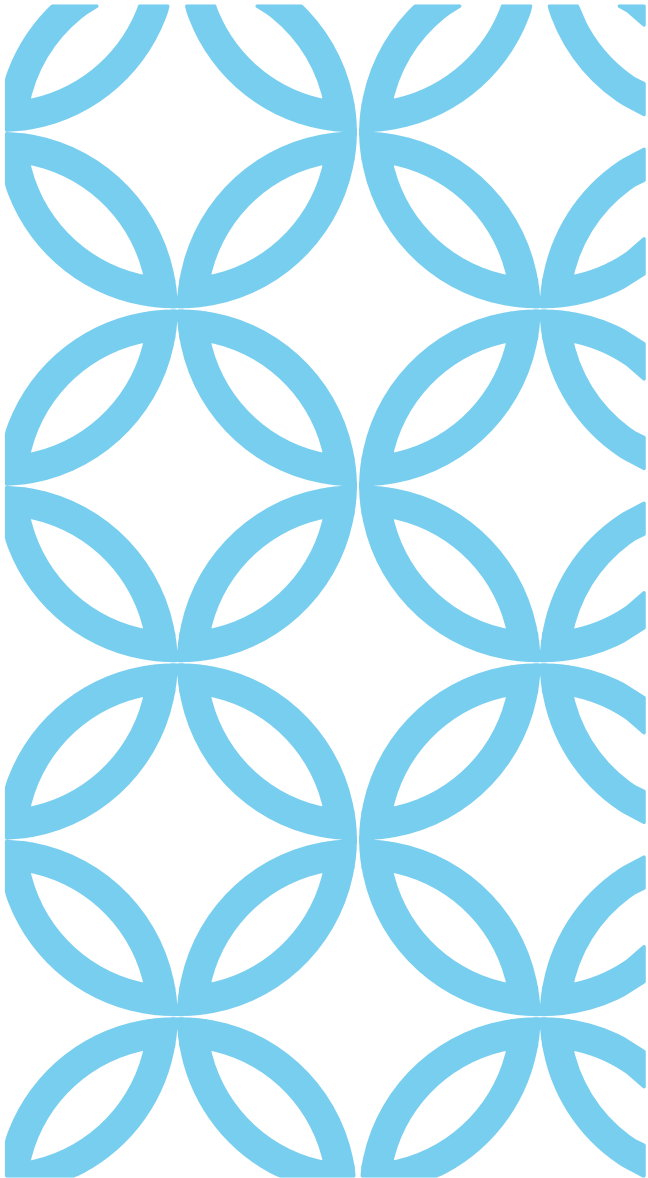
¿ sucedió algo ?

No pasó nada, ya que sólo entrenamos el modelo y lo que necesitamos es pedir que el modelo **prediga** quién es el elemento misterioso, utilizando el método *predict*:

```
modelo.predict(misterioso)
```

Este método devuelve si es un perro o un cerdo, entonces necesitamos imprimir con la sentencia *print*:

SKLEARN



Al correr el devolvió un warning(después veremos como resolverlo), imprimió un resultado que es un array ... [-1] ¿ qué significa -1 para el algoritmo ?.

Significa que es un perro, ¿ cómo dedujo que era un perro ?

Veamos nuestro código ...

SKLEARN

```
pi@raspberrypiML2:~/ML $ python3 clasificacion1 3.py
```

```
/usr/lib/python3/dist-packages/sklearn/utils/validation.py:395:
```

```
DeprecationWarning: Passing 1d arrays as data is deprecated in 0.17 and will raise  
ValueError in 0.19. Reshape your data either using X.reshape(-1, 1) if your data has  
a single feature or X.reshape(1, -1) if it contains a single sample.
```

```
DeprecationWarning)
```

```
[-1]
```

```
pi@raspberrypiML2:~/ML $
```

Veamos nuestro código ...



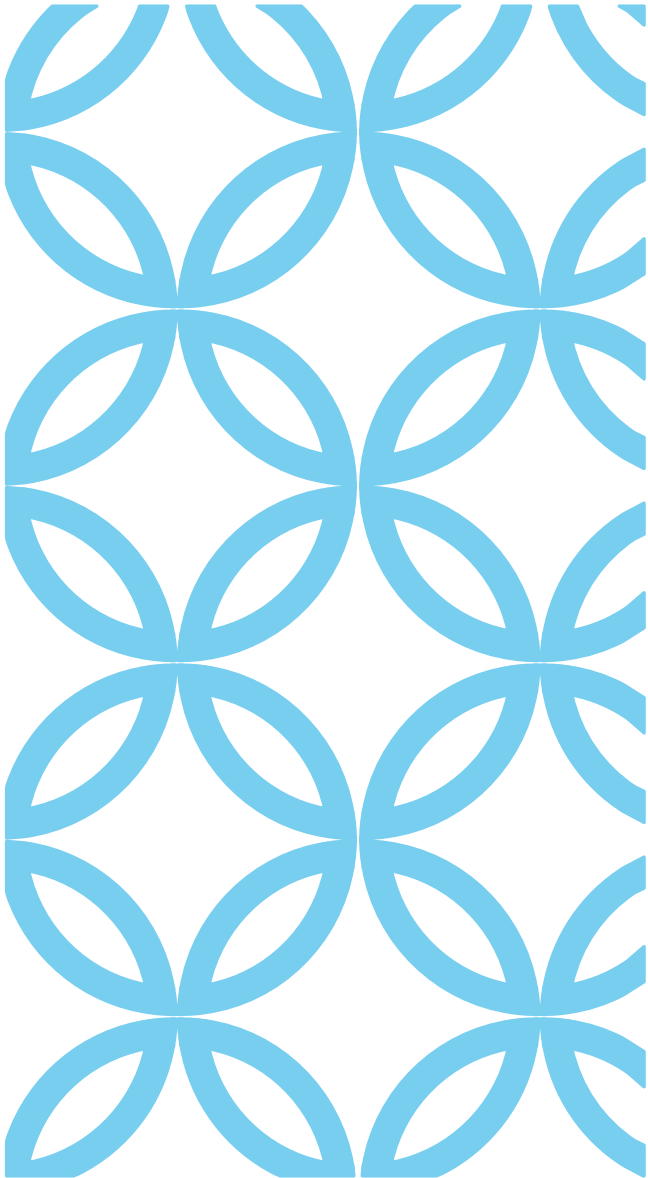
PYTHON

```
# [ es gordito ? Tiene pierna corta? Dice guau guau ?]
cerdo1 = [ 1, 1, 0 ]
cerdo2 = [ 1, 1, 0 ]
cerdo3 = [ 1, 1, 0 ]
perro1 = [ 1, 1, 1 ]
perro2 = [ 1, 0, 1 ]
perro3 = [ 1, 0, 1 ]

datos = [cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
etiquetas = [1,1,1,-1,-1,-1]

misterioso = [ 1, 1, 1 ]

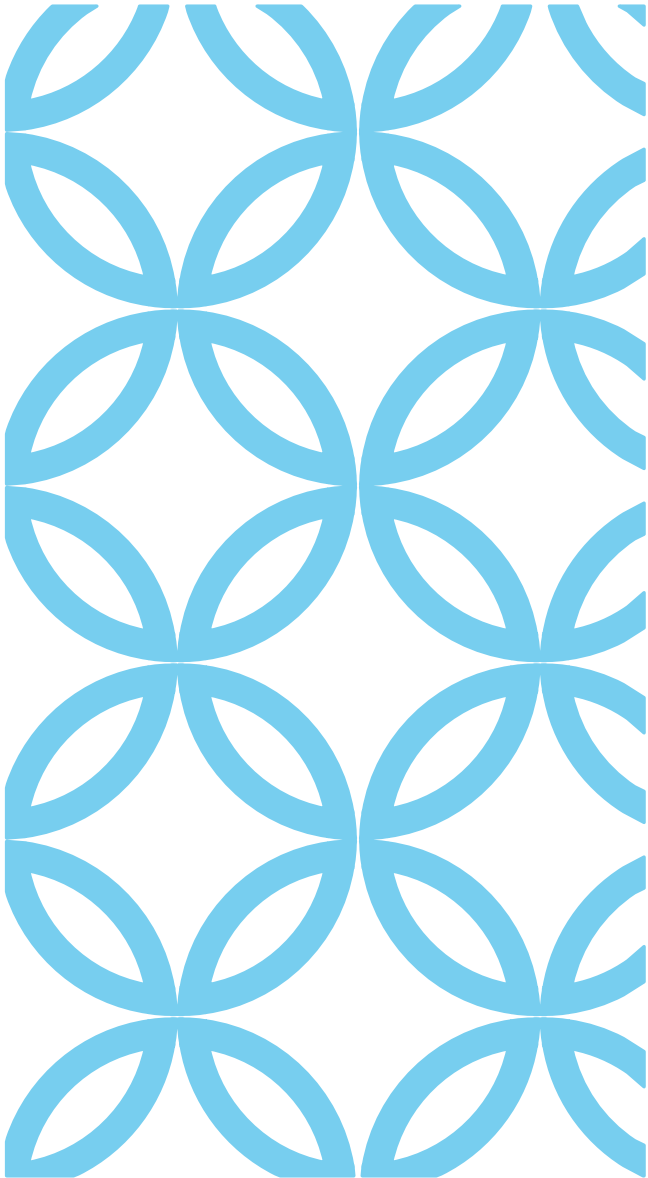
from sklearn.naive_bayes import MultinomialNB
modelo = MultinomialNB()
modelo.fit(datos, etiquetas)
print(modelo.predict(misterioso))
```



Notar que pasamos varios datos para nuestro programa, entrenamos y le pedimos verificar un único caso que fue el elemento misterioso ...

¿ Será que queremos predecir apenas un solo caso ?

SKLEARN



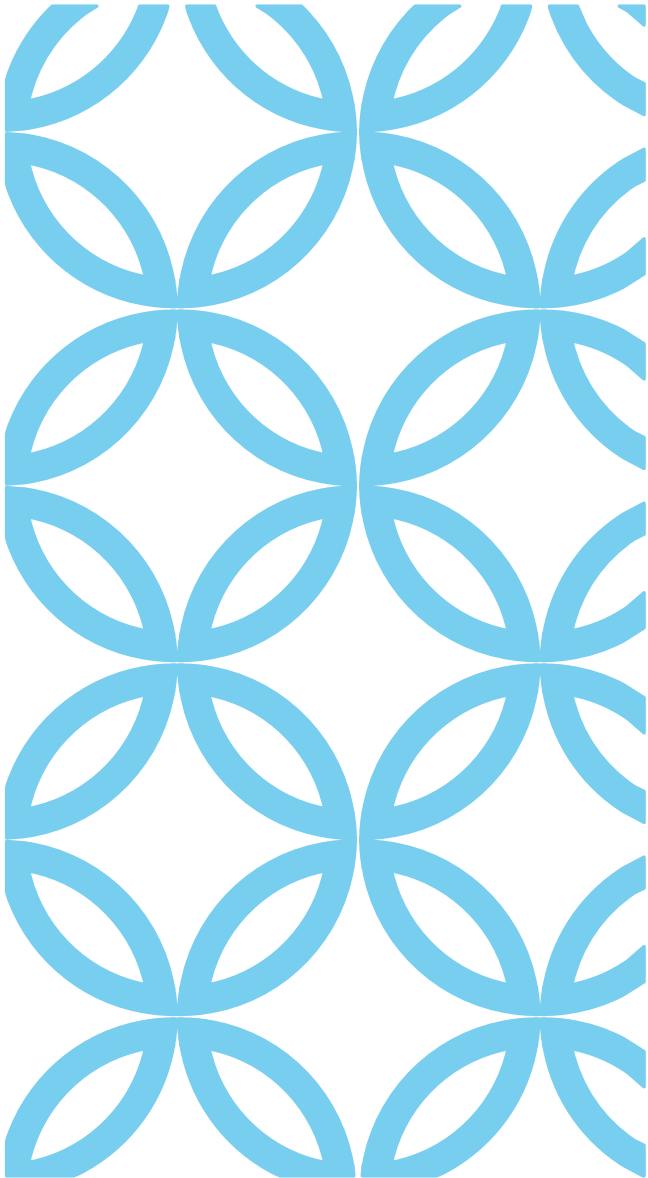
Entonces necesitamos de otros elementos misteriosos para que el programa clasifique por nosotros.

Por ejemplo tenemos el misterioso2 que es gordo, no tiene pierna corta y no dice guau gau.

misterioso1 = [1, 1, 1]

misterioso2 = [1, 0, 0]

SKLEARN



Ahora en vez de predecir el misterioso1, pediremos que prediga el misterioso2:

```
print (modelo.predict(misterioso2))
```

... vemos nuestro código:

SKLEARN



PYTHON

```
# [ es gordito ? Tiene pierna corta? Dice guau guau ?]
cerdo1 = [ 1, 1, 0 ]
cerdo2 = [ 1, 1, 0 ]
cerdo3 = [ 1, 1, 0 ]
perro1 = [ 1, 1, 1 ]
perro2 = [ 1, 0, 1 ]
perro3 = [ 1, 0, 1 ]

datos = [cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
etiquetas = [1,1,1,-1,-1,-1]

misterioso1 = [ 1, 1, 1 ]
misterioso2 = [ 1, 0, 0 ]

from sklearn.naive_bayes import MultinomialNB
modelo = MultinomialNB()
modelo.fit(datos, etiquetas)
print(modelo.predict(misterioso2))
```

Corremos nuevamente nuestro código:

```
pi@raspberrypiML2:~ $ cd ML
```

```
pi@raspberrypiML2:~/ML $ python3 clasificacion.py
```

```
/usr/lib/python3/dist-packages/sklearn/utils/validation.py:395:
```

```
DeprecationWarning: Passing 1d arrays as data is deprecated in 0.17 and will raise  
ValueError in 0.19. Reshape your data either using X.reshape(-1, 1) if your data has  
a single feature or X.reshape(1, -1) if it contains a single sample.
```

```
DeprecationWarning)
```

```
[1]
```

```
pi@raspberrypiML2:~/ML $
```

```
=====
```

La interpretación que da el algoritmo con mayor opción al misterioso2 es que sea un cerdo. Ahora si queremos predecir todos los misteriosos de una sola vez necesitamos adicionar todos los misteriosos dentro de un array y asignar a la variable testeo.

Variable testeo:

```
misterioso1 = [ 1, 1, 1 ]
```

```
misterioso2 = [ 1, 0, 0 ]
```

```
testeo = [ misterioso1, misterioso2 ]
```

Y pasamos teste en el argumento de la función predict:

```
print( modelo.predict( testeo ) )
```

Veamos como queda nuestro código...



PYTHON

```
# [ es gordito ? Tiene pierna corta? Dice guau guau ?]
```

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

```
perro3 = [ 1, 0, 1 ]
```

```
datos = [cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
```

```
etiquetas = [1,1,1,-1,-1,-1]
```

```
misterioso1 = [ 1, 1, 1 ]
```

```
misterioso2 = [ 1, 0, 0 ]
```

```
testeo = [misterioso1, misterioso2 ]
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
modelo = MultinomialNB()
```

```
modelo.fit(datos, etiquetas)
```

```
print(modelo.predict(testeo))
```

¿ vamos a correr nuestro código nuevamente ?

```
pi@raspberrypiML2:~/ML $ python3 clasificacion.py
```

```
[-1  1]
```

```
pi@raspberrypiML2:~/ML $
```

Ahora imprime dos resultados y muestra que el primer elemento misterioso es un perro y el segundo es un cerdo. Si verificamos nuestros dos elementos misteriosos:

```
misterioso1 = [ 1, 1, 1 ]
```

```
misterioso2 = [ 1, 0, 0 ]
```

Realmente el misterioso1 es un perro y el misterioso2 es un cerdo.



PYTHON

¿ entonces cómo
funciona el algoritmo
Bayesiano Multinomial
para clasificación ?

- entrenamos nuestro modelo pidiendo que se adecúe a los datos y etiquetas utilizando el método fit.
- Pedimos que prediga que son por medio del método predict.

Así el algoritmo va prediciendo si esos elementos son cerdos o canes.

¿ Será que el código que creamos sólo funciona para clasificar cerdos y perros ?

¿ Si en vez de cerdos y canes los elementos fuesen spam y no spam ?

¿ Y si fuese que un cliente paga o no paga las deudas ?

¿ Y si esos 0s y 1s clasifican cualquier otra cosa ?

¿ cambiaría alguna cosa en el código ?

¿ Qué creen ustedes ?

Tomando este ejemplo podemos implementar la misma cosa que vimos con tablas y números y traducir esto en código. Es bastante sencillo ... ¿ Cuáles son los pasos ?

- Para cada uno de los elementos definimos la variables, por ejemplo: `cerdo1 = [1,1,0]` , `cerdo2=[1,1,0]` y así por delante.
- Cada variable tiene un significado, en nuestro ejemplo fue: [¿ es gordo ?, ¿ tiene pierna corta ?, ¿ dice guau guau?].
- Etiquetamos esos elementos para diferenciarlos uno de otros, en nuestro ejemplo utilizamos 1 para cerdo y -1 para perro: `etiquetas=[1,1,1,-1,-1,-1]`.
- Luego entrenamos nuestro modelo: `modelo.fit(datos, etiquetas)`.
- Y por último pedimos imprimir nuestras pruebas: `print(modelo.predict(testes))`.

Notar que las etiquetas fueron hechas de una forma diferente que vimos en la tabla: en vez de usar 0 y 1, usamos -1 y 1. Optamos por esa opción, pues da un énfasis de mejor distinción. Podríamos hacer con 0 y 1 sin problema alguno:

`Etiquetas=[1, 1, 1, 0, 0, 0]`



PYTHON

Otro ejemplo que podemos citar es la música que está siendo reproducida en una plataforma. Necesitamos saber si es pirata o no.

Tenemos que verificar si el usuario está practicando piratería y si él tiene los derechos para bajar el contenido o no.

Todo eso significa si que precisamos clasificar basado en las características que tenemos y que para cada caso usaremos características (features) diferentes.



PYTHON

Algo que es importante y común a la vez en todos los algoritmos que utilizamos como también para nosotros: ***no siempre tenemos la certeza de nuestra clasificación.***

Tenemos que considerar que todo algoritmo contiene un margen de error, luego es necesario contemplar que se puede dar un margen de error en nuestro algoritmo.

Es importante estar atento a los errores que se puedan presentar para poder controlarlos. Es necesario conocer nuestro margen de error y comprender cual es el valor que vamos a aceptar en nuestro algoritmo.



¿ ES UN GATO ?

Vamos poner atención al margen de error y aprender como calcularlo para decir si ese error es aceptable o si estamos mejorando nuestro margen actual.

Podemos concluir que todos nuestros algoritmos presentarán errores, pero nuestro papel es a partir de ellos, verificar si son aceptables o no.

Por ejemplo, ¿es aceptable que mi foto sea clasificada como la de un gato?. No existe una regla fija para saber si eso es aceptable o no, dependerá del contexto para decir si nuestro error es aceptable o no; si necesitamos un margen de error menor, o si está dentro del patrón que esperamos.

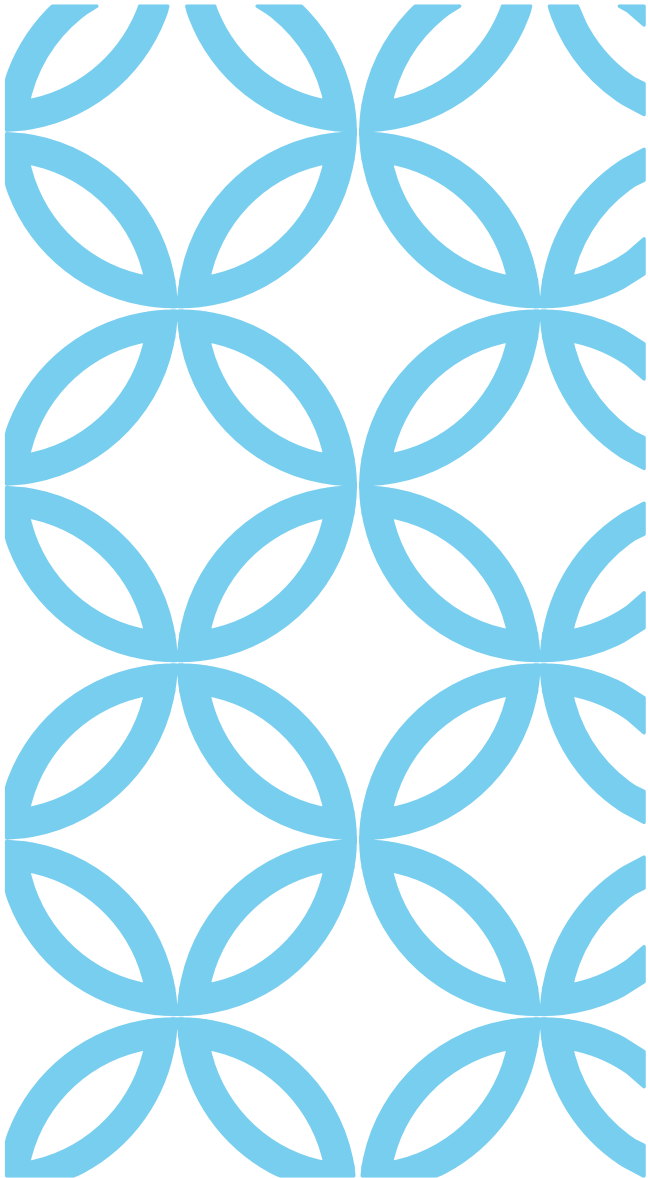


PYTHON

Existe un detalle en el algoritmo que creamos, no adelanta correr nuestro algoritmo y no hacernos la idea de cuan bueno es. Cuando entrenamos nuestro cerebro, podemos llegar a ver 300 o más, cerdos y perros en nuestra vida, pero no tenemos la certeza de que estamos bien lo suficiente para diferenciar entre un cerdo y un can, o entonces spam y no spam, o deudores que van a pagar o no pagar.

¿Cómo podemos tener la certeza si estamos bien o no para hacer esas distinciones?

Necesitamos testear (verificar)!!!. Sí, eso probar en el mundo real con algún valor que desconocemos y ver si va a funcionar conforme a lo esperado para verificar cuán bien estamos con el algoritmo.



¿ y cómo podemos verificar ?

SKLEARN



PYTHON

Podemos crear un escenario de pruebas (testear) simulando los elementos reales (cerdos y perros) por elementos misteriosos:

```
misterioso1 = [1, 1, 1]
```

```
misterioso2 = [1, 0, 0]
```

```
misterioso3 = [0, 0, 1]
```

```
testeo = [misterioso1, misterioso2, misterioso3]
```

Ahora que tenemos nuestro escenario de pruebas, podemos comparar los resultados esperados para cada uno de esos elementos misteriosos, o sea etiquetas de testeo.

```
etiquetas_testeo
```



PYTHON

Ahora necesitamos clasificar cada elemento misterioso e indicar en nuestra etiquetas de testeo.

Sabemos que misterioso1 es un perro (-1), misterioso2 es un cerdo (1) y misterioso3 es un perro (-1).

```
etiquetas_testeo = [ -1, 1, -1 ]
```

Para quedar mas claro vamos asignar el retorno del método ***predict()*** a una variable que llamaremos ***resultado***:

```
resultado = modelo.predict(testeo)  
print(resultado)
```

Notar como queda nuestro código:



PYTHON

```
# [ es gordito ? Tiene pierna corta? Dice guau guau ?]
```

```
cerdo1 = [ 1, 1, 0 ]
```

```
cerdo2 = [ 1, 1, 0 ]
```

```
cerdo3 = [ 1, 1, 0 ]
```

```
perro1 = [ 1, 1, 1 ]
```

```
perro2 = [ 1, 0, 1 ]
```

```
perro3 = [ 1, 0, 1 ]
```

```
datos = [cerdo1,cerdo2,cerdo3,perro1,perro2,perro3]
```

```
etiquetas = [1,1,1,-1,-1,-1]
```

```
from sklearn.naive_bayes import MultinomialNB
```

```
modelo = MultinomialNB()
```

```
modelo.fit(datos, etiquetas)
```

```
misterioso1 = [ 1, 1, 1 ]
```

```
misterioso2 = [ 1, 0, 0 ]
```

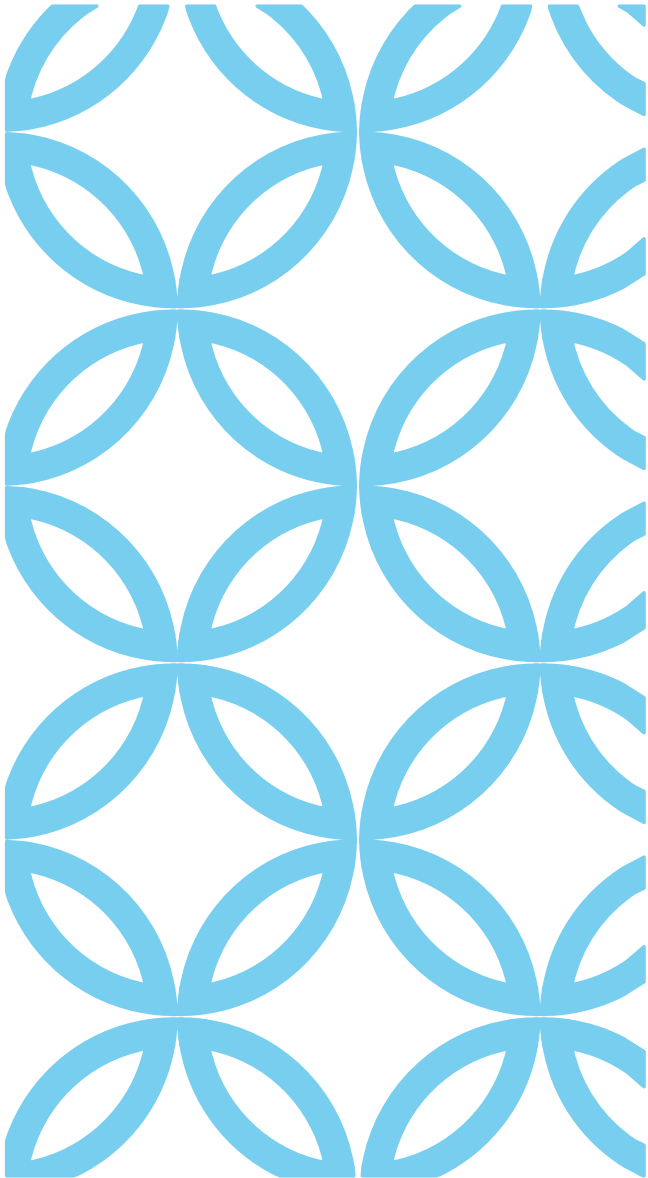
```
misterioso3 = [ 0, 0, 1 ]
```

```
testeo = [misterioso1, misterioso2, misterioso3 ]
```

```
etiquetas_testeo = [ -1, 1, -1 ]
```

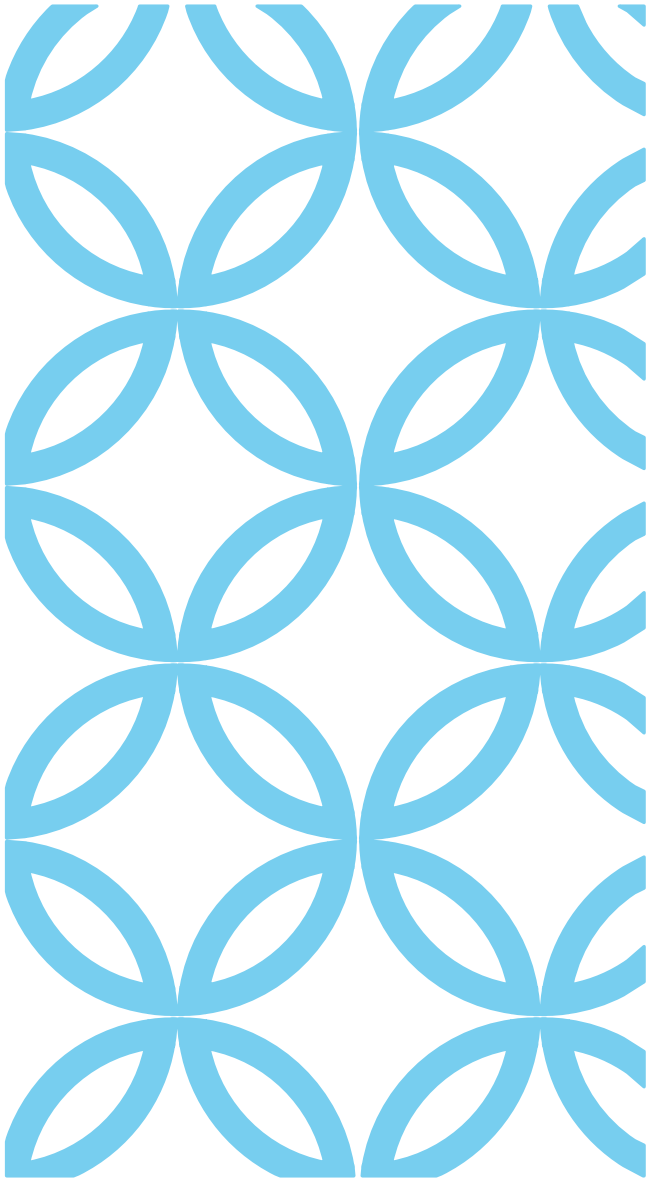
```
resultado = modelo.predict(testeo)
```

```
print(resultado)
```

¿ será que nuestro código salió bien ?

SKLEARN



esperando en nuestra etiquetas de testeo son los correctos:

```
etiquetas_testeo = [ -1, 1, -1 ]
```

De acuerdo con nuestras etiquetas, el código funcionó muy bien, más es importante notar que en el mundo real (en la práctica), este resultado no acostumbra ser 100%.

SKLEARN



PYTHON

... Pero; observe que estamos testeando eso manualmente, estamos verificando si el testeo pasó a simple vista. Para nuestro caso no hay problema, más en el mundo real, realizaremos testeos gigantes, como por ejemplo con 1.000 o más elementos y no tiene sentido que nos quedemos observando uno por uno para verificar si acertó o erró.

Lo que necesitamos saber es cuántos elementos de la variable resultado (resultados que el algoritmo clasificó) son diferentes de la variable etiquetas_testeo (resultado esperado de los elementos).

¿ Más cómo podemos verificar si los valores son diferentes?

Podemos sustraer los dos arrays:

```
print( resultado – etiquetas_teseo)
```



PYTHON

... Resumiendo para todas las posibilidades que hicimos en los casos en que los valores fueran iguales el resultado será 0. Dicho de otra manera, cuando resulte en 0 sabremos que el algoritmo acertó.

Vamos a incorporar la siguiente línea a nuestro código:

```
print(resultado - etiquetas_testeo)
```

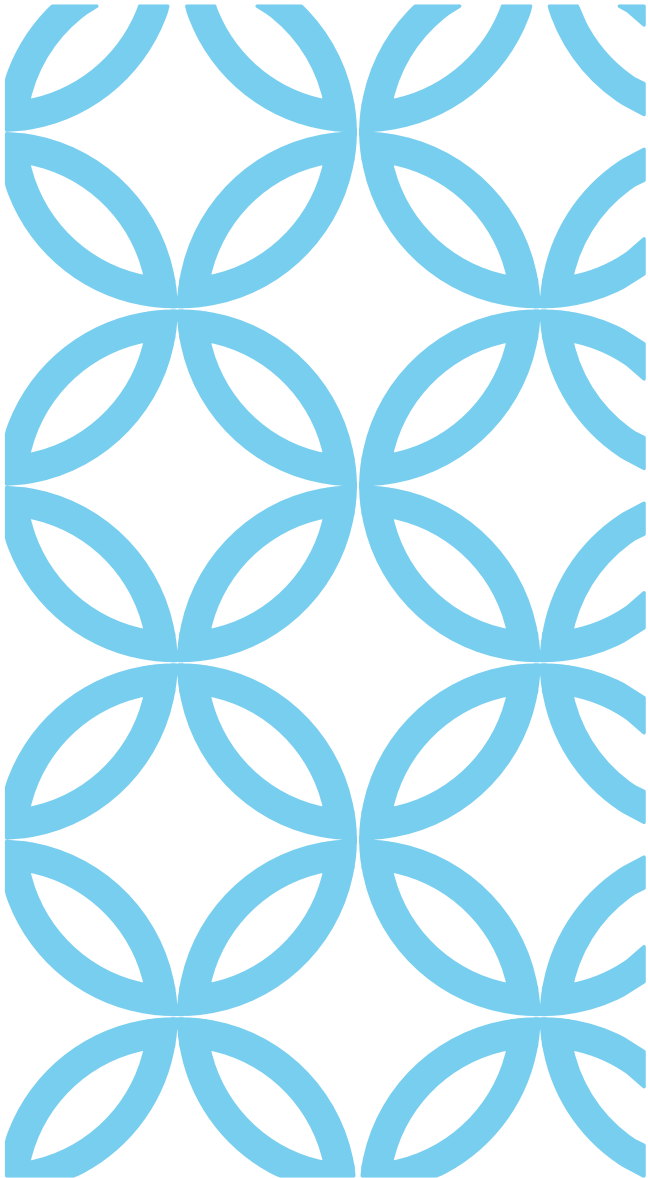
Y hacemos correr nuestro código.

```
pi@raspberrypiML2:~/ML $ python3 clasificacion.py
```

```
[-1  1 -1]
```

```
[0 0 0]
```

```
pi@raspberrypiML2:~/ML $
```



El resultado fue todo 0, entonces acertó 100% nuevamente, lo ideal es que el algoritmo muestre el porcentaje en vez de solo la diferencia entre **resultado** y **etiquetas_testeo**. Vamos a calcular la **diferencia** entre las dos variables.

$\text{diferencias} = \text{resultado} - \text{etiquetas_testeo}$

Vamos a calcular la tasa de aciertos(precisión,accuracy).

SKLEARN



PYTHON

...

```
diferencias = resultado - etiquetas_testeo
```

```
aciertos = [ d for d in diferencias if d==0 ]
```

```
total_aciertos = len(aciertos)
```

```
total_elementos = len(testeo)
```

```
tasa_acierto=100.0*total_aciertos/total_elementos
```

```
print('resultado: ',resultado)
```

```
print('diferencias: ', diferencias)
```

```
print('tasa aciertos: ', tasa_aciertos,'%')
```

Corriendo nuestro código tenemos:

```
pi@raspberrypiML2:~/ML $ python3 clasificacion.py
```

```
resultado: [-1  1 -1]
```

```
diferencias: [0 0 0]
```

```
tasa de aciertos: 100.0 %
```

```
pi@raspberrypiML2:~/ML $
```

... 100% pero ... siempre es bueno recordar que 100.0 es un número muy difícil que se pueda dar en el mundo real. Un ejemplo bien simple que demuestra que nuestro algoritmo erra, sería modificar los valores de nuestra variable ***etiquetas_testeo***, haciendo que el último elemento misterioso era un cerdo que decía guau guau:

```
etiquetas_testeo = [ -1, 1, 1 ]
```



PYTHON

¿ Vamos a verificar si nuestro algoritmo va “adivinar” que ese cerdo que dice guau guau es un cerdo ?

```
pi@raspberrypiML2:~/ML $ python3  
clasificacion19.py
```

```
resultado: [-1  1 -1]
```

```
diferencias: [ 0  0 -2]
```

```
tasa de aciertos: 66.666666666666667 %
```

```
pi@raspberrypiML2:~/ML $
```




PYTHON

Como vemos, el algoritmo erró, notar que el último valor de la diferencia es distinto de 0, o sea un error. Por último nuestra tasa de acierto fue de un 66%.

Repare que la tasa de acierto es fundamental para que nuestro algoritmo en el momento que recibe nuevos datos consiga verificar cuán bueno es en una situación del mundo real. Como por ejemplo en esta última prueba fue capaz de acertar 66% al tener que clasificar un nuevo animal que poseía características inesperadas.



...Contenido del curso en Desarrollo
... continuará