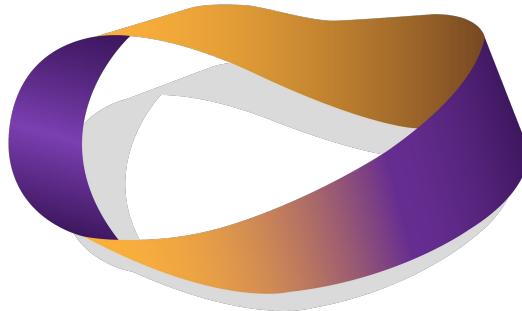


# Cómputo cognitivo a través de IBM Watson



**ACTUMLOGOS**

DESARROLLANDO HABILIDADES TECNOLÓGICAS

## Introducción: IBM Watson y cómputo cognitivo

En la historia de la inteligencia artificial se ha intentado, en varias ocasiones, ponerla a prueba. En una de esas ocasiones, Watson consiguió ganar a los dos mejores jugadores del popular programa Jeopardy. Lo que hizo Watson fue ejecutar cientos de algoritmos de análisis de lenguaje para encontrar las respuestas correctas en 200 millones de páginas de contenido (incluido Wikipedia).

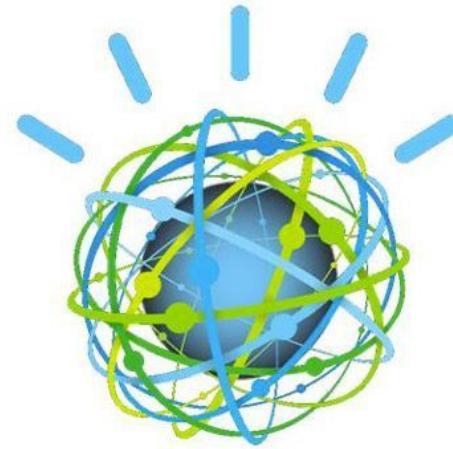


Los investigadores de IBM entrenaron a Watson usando técnicas de aprendizaje automático (machine learning) y aprendizaje por reforzamiento (reinforcement learning). En los últimos años la popularidad de este sistema se ha incrementado y se pueden encontrar cientos de artículos y documentación sobre el funcionamiento de esta tecnología.



*Hola, me llamo Watson.*

IBM Watson es una plataforma de cómputo cognitivo basada en la nube, que ha sido ampliamente utilizada para resolver problemas del mundo real. Los sistemas cognitivos buscan simular el comportamiento y la toma de decisiones como lo haría un ser humano.



**IBM** **Watson**

## Cuenta en la Nube de IBM

Vamos a necesitar una cuenta gratuita de la Nube de IBM (IBM Cloud) para acceder a los servicios de nivel *Watson Lite* (este tipo de acceso es limitado, sin embargo es suficiente para familiarizarse con las características de Watson y empezar el desarrollo de aplicaciones).

Para obtener una cuenta gratuita de IBM Cloud debemos seguir las instrucciones dadas en:

<https://cloud.ibm.com/docs/services/watson?topic=watson-about#about>

### Iniciación a Watson y IBM Cloud

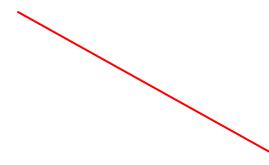
Última actualización: 2019-06-04

Está a pocos pasos de empezar con IBM Watson™.

Iniciar  
registro

#### Paso 1: Obtener una cuenta gratuita de IBM Cloud

Cree una cuenta en IBM Cloud para probar los servicios de Watson de forma gratuita sin restricciones de tiempo: [Registrarse de forma gratuita](#). Recibirá un correo electrónico para confirmar y activar su cuenta.



Se nos solicita una cuenta de correo a donde se envía un código de verificación válido por 30 min.

The screenshot shows the IBM Cloud account creation interface. On the left, there's a sidebar with the 'IBM Cloud' logo and navigation links for 'Catalog' and 'Docs'. Below that, it asks if you have an account and provides a 'Iniciar sesión' link. The main area is titled 'Crear una cuenta' (Create an account). It consists of two stacked sections: '1. Información de la cuenta' (Account information) and '2. Verificar correo electrónico' (Verify email address). The first section has a green checkmark and an 'Editar' button. The second section contains a message about an email verification code sent to 'jmvn2@hotmail.com'. It includes a 'Código de verificación' input field with a red border and a red exclamation mark, and a message below stating 'Código de verificación es obligatorio.' (Verification code is required). At the bottom are 'Siguiente' and 'Reenviar código' buttons. To the right of the form is a large promotional banner with the text 'Build for free on IBM Cloud', 'Develop for free, no credit card required', 'Access the full catalog at your fingertips', and descriptive text about apps, AI, and analytics.

IBM Cloud

Catalog Docs

¿Ya dispone de una cuenta de IBM Cloud? [Iniciar sesión](#)

## Crear una cuenta

1. Información de la cuenta ✓ [Editar](#)

2. Verificar correo electrónico

Hemos enviado un código de verificación de 7 dígitos a [jmvn2@hotmail.com](mailto:jmvn2@hotmail.com). Este código es válido durante 30 minutos. ¿Desea crear esta cuenta con otro correo electrónico? [Editar correo electrónico](#).

Código de verificación

Código de verificación es obligatorio.

Siguiente Reenviar código

Build for free  
on IBM Cloud

Develop for free, no credit card required

Apps, AI, analytics, and more. Build with 40+ Lite plan services at no cost to you - ever.

Access the full catalog at your fingertips

Upgrade your account and unlock 190+ unique offerings, plus get a \$200 credit to use with any offering you want.



Hello,

Thank you for signing up for IBM Cloud!

Your 7-digit verification code is:

**8400169**

Enter this verification code on the IBM Cloud registration page where you requested the code. This code is valid for 30 minutes.

Welcome and happy building!

Thank you,  
IBM Cloud

Visit the [IBM Cloud console](#).

© Copyright IBM Corporation 2014, 2020.



Se llena los datos solicitados y aceptan las condiciones de IBM.

## 2. Verificar correo electrónico ✓

[Editar](#)

Su dirección de correo electrónico se ha verificado.

## 3. Información personal ✓

[Editar](#)

Cuenta personal

Jesús Manuel Vázquez

México

[Crear cuenta](#)



IBM may use my contact data to keep me informed of products, services and offerings:

by email.

by telephone.

### Your rights

Our [Privacy Statement](#) provides more information about your personal data rights. It also provides contact information if you have questions or concerns regarding our handling of your personal data.

### Acknowledgement

I acknowledge that I understand how IBM is using my Basic Personal Data and I am at least 16 years of age.

[Proceed](#)

[Cancel Sign In](#)



Hi Jesús Manuel,

Thanks for creating an IBM Cloud account! Your account is set up and ready for you. Log in to start building today.

[Log in](#)

Thank you,  
IBM Cloud

Visit the [IBM Cloud console](#).

© Copyright IBM Corporation 2014, 2020.



Una vez que se termina el registro, se inicia sesión para entrar al tablero de Watson (*Watson dashboard*), en donde se puede realizar las siguientes acciones:

- Explorar los servicios de Watson.
- Enlace a los servicios que ya ha registrado para usar.
- Vistazo a los recursos de desarrollador que incluyen la documentación de Watson, SDK's y recursos para aprender sobre Watson.
- Ver las aplicaciones que ha creado con Watson.

## Servicios de Watson

### 1. Asistente de Watson (**Watson Assistant**)

Este servicio permite crear *chatbots* y asistente virtuales que permiten a los usuarios interactuar con texto en lenguaje natural. IBM provee una interfaz web que se puede utilizar para *entrenar* el asistente de Watson para diferentes escenarios asociados con las aplicaciones.

### 2. Reconocimiento Visual (**Visual Recognition**)

Este servicio permite a la aplicación localizar y entender información en imágenes y video incluyendo colores, objetos, rostros, texto, comida y contenido inapropiado. Existen modelos pre-entrenados o se pueden entrenar los propios.

### 3. Voz a Texto (**Speech to Text**)

Permite convertir archivos de audio de voz a sus respectivas transcripciones de texto. Este servicio puede ser utilizado para implementar aplicaciones que sean controladas por voz, transcribir audio en vivo, etc.

#### **4. Texto a Voz (**Text to Speech**)**

Permite convertir texto a una voz sintetizada. Actualmente, este servicio soporta idiomas como el Inglés, Francés, Alemán, Italiano, Español, Portugues y Japones.

#### **5. Traductor (**Language Translator**)**

Este servicio permite la traducción de texto entre lenguajes e identificar el lenguaje en el cual está escrito un texto.

#### **6. Comprensión del Lenguaje Natural (**Natural Language Understanding**)**

Analiza un texto y produce información que incluye el sentimiento y la emoción. Este servicio también permite identificar personas, lugares, organizaciones, compañías y cantidades. También permite obtener categorías y conceptos como deportes, gobiernos y políticos.

#### **7. Descubrimiento (**Watson Discovery**)**

Este servicio permite a las empresas estructurar y comprender, con cierta facilidad, la gran masa de datos de las que disponen, en muchas ocasiones de fuentes muy dispares como lo son documentos de aplicaciones de ofimática, documentos HTML, formato JSON o formato PDF, etc.

## SDK de desarrollo en Watson con Python

Para el desarrollo de proyecto utilizando Watson, IBM brinda el SDK de desarrollo en Watson con Python (Watson Developer Cloud Python SDK), que contiene las clases necesarias para interactuar con los servicios de Watson. Vamos a crear objetos para cada servicio e interactuar a través de los métodos de dichos objetos.

Los usuarios de Windows requiere instalar Microsoft C++ build tools

<https://visualstudio.microsoft.com/es/visual-cpp-build-tools/>

Posteriormente para instalar el SDK ejecutamos

```
pip install --upgrade watson-developer-cloud
```

Para el caso práctico que vamos a desarrollar, también se necesita instalar las siguientes bibliotecas.

```
conda install -c conda-forge pydub  
conda install -c conda-forge ffmpeg  
conda install -c anaconda pyaudio  
conda install portaudio
```

## Caso de estudio: Compañero de viaje para traducción

Imagina que estás viajando a un país donde no hablan español y no puedes comunicarte porque no hablas el idioma del país. En ese momento, una aplicación para la traducción se vuelve necesaria para poder comunicarte.



Para realizar esta aplicación vamos a utilizar tres servicios de Watson que permitan el desarrollo de este caso de estudio:

- *Speech to Text*
- *Text to Speech*
- *Language Translator*

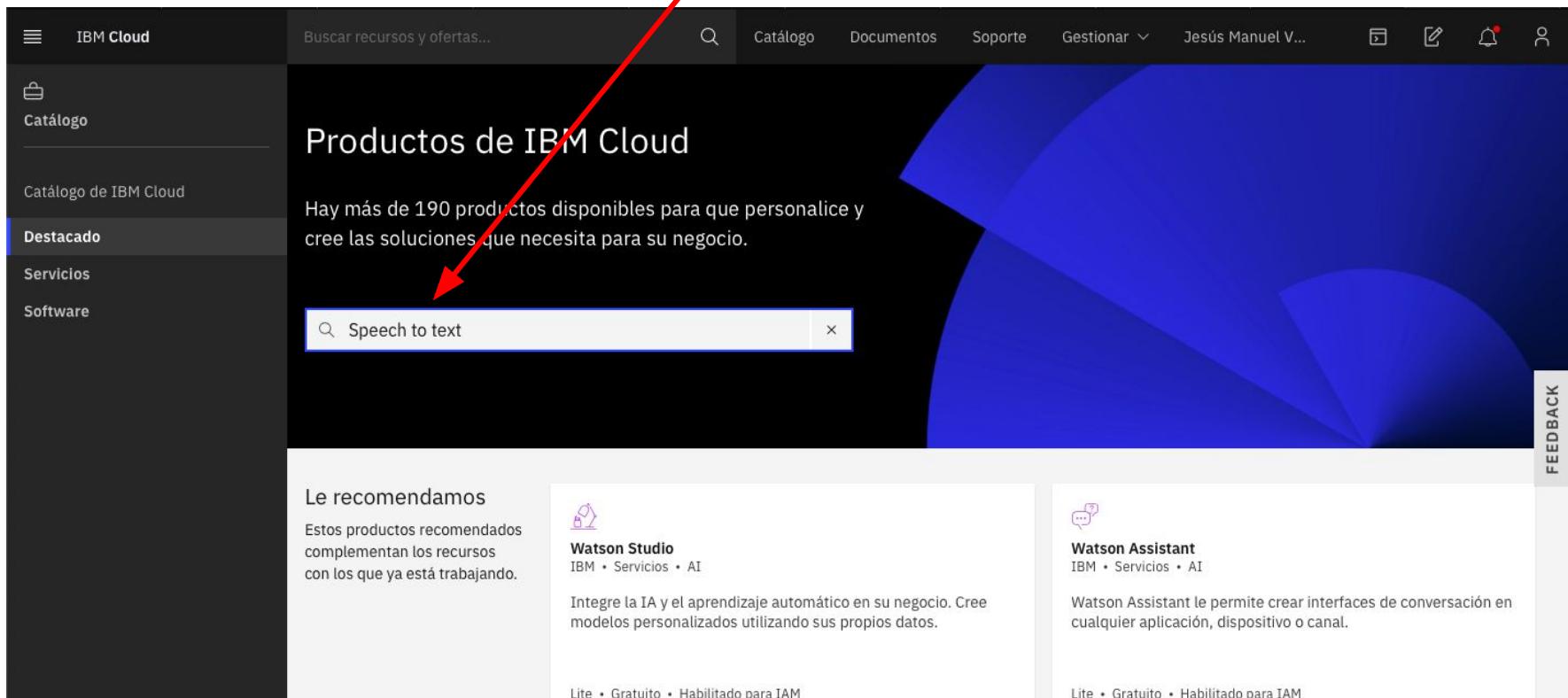
Para cada uno de estos servicios, se debe crear un servicio y obtener sus credenciales que nos permitan interactuar, como lo hicimos en su momento con Twitter.



Nos dirigimos a la nube de IBM: <https://cloud.ibm.com> y creamos un nuevo recurso

The screenshot shows the IBM Cloud Control Panel interface. At the top, there is a navigation bar with links for Catalog, Docs, Support, Manage, and a user profile. A red arrow points from the text above to the 'Crear recurso' (Create resource) button in the top right corner of the main content area. The main content area is titled 'Panel de control' and contains several sections: 'Resource summary' (with a 'View all' link), 'Planned maintenance' (with a 'View all' link), 'For you' (listing 'Watson Studio' and 'Iniciación a Watson Studio'), 'News' (mentioning 'we.trade Digital Trade Finance Network strengthens collaboration with IBM'), 'Recent support cases' (with a 'View all' link), 'User access' (with a 'Manage users' link), and 'IBM Cloud status' (with a 'View all' link). On the far left, there is a sidebar with various icons and a '+' sign. On the far right, there is a vertical 'FEEDBACK' bar.

En el catálogo de productos de IBM, buscamos el servicio deseado. En este caso buscamos el servicio *Speech to text*.



The screenshot shows the IBM Cloud Catalog interface. On the left, there's a sidebar with navigation links: 'IBM Cloud', 'Catálogo' (which is selected), 'Catálogo de IBM Cloud', 'Destacado' (selected), 'Servicios', and 'Software'. The main content area has a dark background with a large blue abstract graphic on the right. At the top, there's a search bar with the placeholder 'Buscar recursos y ofertas...' and a magnifying glass icon. Below the search bar, the title 'Productos de IBM Cloud' is displayed. A text message says 'Hay más de 190 productos disponibles para que personalice y cree las soluciones que necesita para su negocio.' In the center, there's a search bar containing the text 'Speech to text' with a magnifying glass icon and a close button ('x'). A red arrow points from the top-left towards this search bar. Below the search bar, there's a section titled 'Le recomendamos' with the sub-section 'Watson Studio' and 'Watson Assistant' listed.

IBM Cloud

Buscar recursos y ofertas...

Catálogo

Catálogo de IBM Cloud

**Destacado**

Servicios

Software

Productos de IBM Cloud

Hay más de 190 productos disponibles para que personalice y cree las soluciones que necesita para su negocio.

Speech to text

FEEDBACK

Le recomendamos

Watson Studio

Watson Assistant

Integre la IA y el aprendizaje automático en su negocio. Cree modelos personalizados utilizando sus propios datos.

Watson Assistant le permite crear interfaces de conversación en cualquier aplicación, dispositivo o canal.

Lite • Gratuito • Habilitado para IAM

Lite • Gratuito • Habilitado para IAM

Los servicios de Watson tienen diferentes tipos de planes, entre los que se encuentra el Lite, el cual es gratuito. Seleccionamos ese plan y creamos el servicio.

## Speech to Text

Autor: IBM • Fecha de última actualización: 28/02/2020 • [Documentos](#) • [documentos de API](#)

[Crear](#)      Acerca de

Seleccione una región

Dallas

Seleccione un plan de precios

Los precios mostrados no incluyen impuestos. Los precios mensuales que se muestran son para el país o región: [Estados Unidos](#)

Plan	Características	Tarifas
Lite	500 minutos al mes	Gratis

El plan Lite le inicia con 500 minutos al mes sin ningún coste. Al actualizar a un plan de pago, obtendrá acceso a funciones de personalización.

Los servicios del plan Lite se suprimen tras 30 días de inactividad.

### Resumen

**Speech to Text**      **Gratis**  
Región: Dallas  
Plan: Lite  
Nombre de servicio: Speech to Text-ab  
Grupo de recursos: Default

[Crear](#)

FEEDBACK

Ya que es creado el servicio nos dirigimos a *Gestionar* y copiamos la clave de API, para colocarla en nuestro archivo de *keys\_watson.py*

The screenshot shows the IBM Watson Speech to Text service management interface. On the left, a sidebar lists options: Gestionar (highlighted with a red arrow), Iniciación, Credenciales de servicio, Plan, and Conexiones. The main area displays the 'Speech to Text-ab' resource details: Grupo de recursos: Default, Ubicación: Dallas, and Add tags. Below this is a guide section with 'Guía de aprendizaje de inicio' (selected) and 'Referencia de API'. The 'Credenciales' section contains the 'Clave de API:' field with a redacted value, a 'Descargar' button, a 'Mostrar credenciales' button (highlighted with a red arrow), and a copy icon. The URL is listed as <https://api.us-south.speech-to-text.watson.cloud.ibm.com/instances/22597974-263d-4160-83f6-...>.

Repetimos el proceso para crear los servicios *Text to Speech* y *Language Translator*.

Gestionar

Iniciación

Credenciales de servicio

Plan

Conexiones

Lista de recursos /

## Text to Speech-5k

Grupo de recursos: Default    Ubicación: Dallas    Add tags

⋮

Empiece viendo la guía de aprendizaje

Guía de aprendizaje de inicio Referencia de API

Plan  
Lite

Actualizar

Credenciales

Clave de API:  
..... Mostrar credenciales

URL:  
<https://api.us-south.text-to-speech.watson.cloud.ibm.com/instances/85ac9671-cd33-4056-b68e-1>

Gestionar

Iniciación

Credenciales de servicio

Plan

Conexiones

[Lista de recursos](#) /



## Language Translator-3q

Grupo de recursos: Default

Ubicación: Dallas

Add tags

Empiece viendo la guía de aprendizaje

[Guía de aprendizaje de inicio](#)

[Referencia de API](#)

Plan

Lite

[Actualizar](#)

### Credenciales

[Descargar](#)

[Mostrar credenciales](#)

Clave de API:

.....



URL:

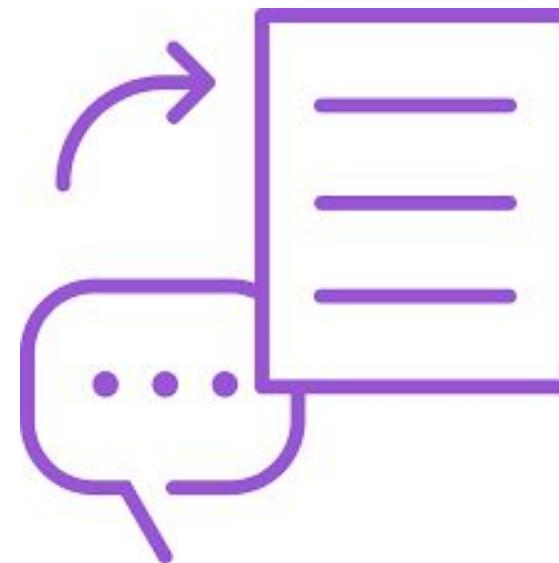
<https://api.us-south.language-translator.watson.cloud.ibm.com/instances/bac98246-ddfc-497e-83d9-1a2f3a23a2c1>



**Reto 1:** En el archivo *Retos\_Clase5\_Watson* y en 10 min, complete las celdas de la 1 a la 4. Estas celdas permiten grabar un audio y utiliza el servicio de *Watson Speech to Text*, para convertir el audio grabado en texto.

**Tips:**

- Consulte en “San Google”
- Importa la clase `SpeechToTextV1` de la biblioteca `watson_developer_cloud`
- Consulte la [documentación](#) para utilizar el modelo de traducción adecuado.



**Resultado esperado:**

Presiona enter para grabar  
Grabando 5 segundos de audio  
Grabacion completa  
hola cómo te llamas

Solución:

```
[ ] # TraductorBasadoEnWatson.py

"""Se utiliza IBM Watson Speech to Text, Language Translator y Text to Speech
APIs para permitir la comunicacion Español/Inglés"""
from watson_developer_cloud import SpeechToTextV1
import keys_watson # llaves para acceder a los servicios de Watson
import pyaudio # utilizado para grabar del microfono
import pydub # utilizados para cargar un archivo WAV
import pydub.playback # utilizado para reproducir el archivo WAV
import wave # utilizado para guardar el archivo WAV

[ ] # Esta celda permite comprobar
# las caracteristicas del microfono
pa = pyaudio.PyAudio()
print(pa.get_device_info_by_index(0))
canales = pa.get_device_info_by_index(0)['maxInputChannels']
```

```
[ ] def grabar_audio(nombre_archivo):
    """Se utiliza pyaudio para grabar 5 segundos en un archivo WAV"""

    #configuraciones generales del audio
    FRAME_RATE = 44100
    CHUNK = 1024
    FORMAT = pyaudio.paInt16
    CHANNELS = 2
    SECONDS = 5

    recorder = pyaudio.PyAudio() # grabación de audio

    # se crean objetos para la captura del audio
    audio_stream = recorder.open(format=FORMAT, channels=CHANNELS,
        rate=FRAME_RATE, input=True, frames_per_buffer=CHUNK)
    audio_frames = [] # objeto para guardar el audio
    print('Grabando 5 segundos de audio')

    # se crea una lista para guarda los datos de audio
    for i in range(0, int(FRAME_RATE * SECONDS / CHUNK)):
        audio_frames.append(audio_stream.read(CHUNK))

    print('Grabacion completa')
    audio_stream.stop_stream() # se termina la grabación
    audio_stream.close()
    recorder.terminate() # se liberan recursos

    # se guardan los datos como archivo WAV
    with wave.open(nombre_archivo, 'wb') as output_file:
        output_file.setnchannels(CHANNELS)
        output_file.setsampwidth(recorder.get_sample_size(FORMAT))
        output_file.setframerate(FRAME_RATE)
        output_file.writeframes(b''.join(audio_frames))
```

```
[ ] audio_esp = 'personal1_español.wav'

input('Presiona enter para grabar')
grabar_audio(audio_esp)

[ ] # objeto para interactuar con Watson Speech to text
stt = SpeechToTextV1(iam_apikey=keys_watson.speech_to_text_key)
model_id = 'es-MX_BroadbandModel' # modelo para la traducción

# se abre el archivo de audio
with open(audio_esp, 'rb') as audio_file:
    # Watson realiza la trascipción del audio
    result = stt.recognize(audio=audio_file, content_type='audio/wav',
        model=model_id).get_result()

# se obtiene la trascipción del audio
texto_esp = result['results'][0]['alternatives'][0]['transcript']
print(texto_esp)
```

**Reto 2:** En el archivo *Retos\_Clase5\_Watson* y en 5 min, complete la celda 5. Se toma el texto producido por el servicio *Watson Speech to Text* en el reto anterior y con el servicio de *Watson Language Translator*, traduce el texto.

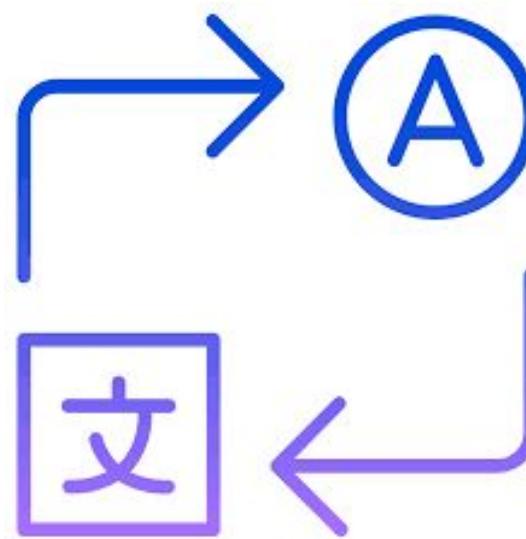
**Tips:**

- Consulte en “San Google”
- Importa la clase `LanguageTranslatorV3` de la biblioteca `watson_developer_cloud`
- Consulte la [documentación](#) para utilizar la versión actual del traductor

**Resultado esperado:**

---

Hi, what's your name?



Solución:

```
[ ] from watson_developer_cloud import LanguageTranslatorV3

# objeto para interactuar con Watson Language Translator
traductor = LanguageTranslatorV3(version='2018-05-31',
                                   iam_apikey=keys_watson.translate_key)

# se realiza la traducción
texto_traducido = traductor.translate(text = texto_esp,
                                         model_id='es-en').get_result()

# se obtiene la lista de traducciones. Si el texto a traducir, tiene
# varias cadenas, la lista trandra multiples entradas. En este caso
# se pasa una sola cadena, por lo que se tiene solo un elemento.
lista_traducciones = texto_traducido['translations'][0]

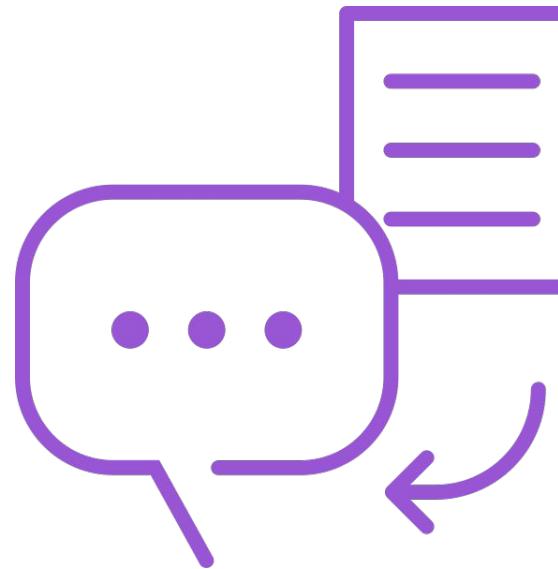
# se obtiene el texto de la traducción
texto_ing = lista_traducciones['translation']

print(texto_ing)
```

**Reto 3:** En el archivo *Retos\_Clase5\_Watson* y en 5 min, complete las celdas 6 y 7 del reto. Se toma el texto producido por el servicio *Watson Speech to Text* en el reto anterior y con el servicio de *Watson Text to Speech*, traduce el texto.

**Tips:**

- Consulte en “San Google”
- Importa la clase `TextToSpeechV1` de la biblioteca `watson_developer_cloud`
- Consulte la [documentación](#) para utilizar la voz adecuada de acuerdo al idioma



**Resultado esperado:**

Audio con la traducción

Solución:

```
[ ] from watson_developer_cloud import TextToSpeechV1

audio_ing = 'personal_1ing.wav'

# objeto para interactuar con Watson Text to Speech
# create Text to Speech client
tts = TextToSpeechV1(iam_apikey=keys_watson.text_to_speech_key)

# abre el documento y graba una voz sintetizada en un archivo
with open(audio_ing, 'wb') as audio_file:
    audio_file.write(tts.synthesize(texto_ing,
        accept='audio/wav', voice='en-US_KevinV3Voice').get_result().content)

[ ] # se utiliza el módulo pydub para reproducir el archivo WAV
sound = pydub.AudioSegment.from_wav(audio_ing)
pydub.playback.play(sound)
```

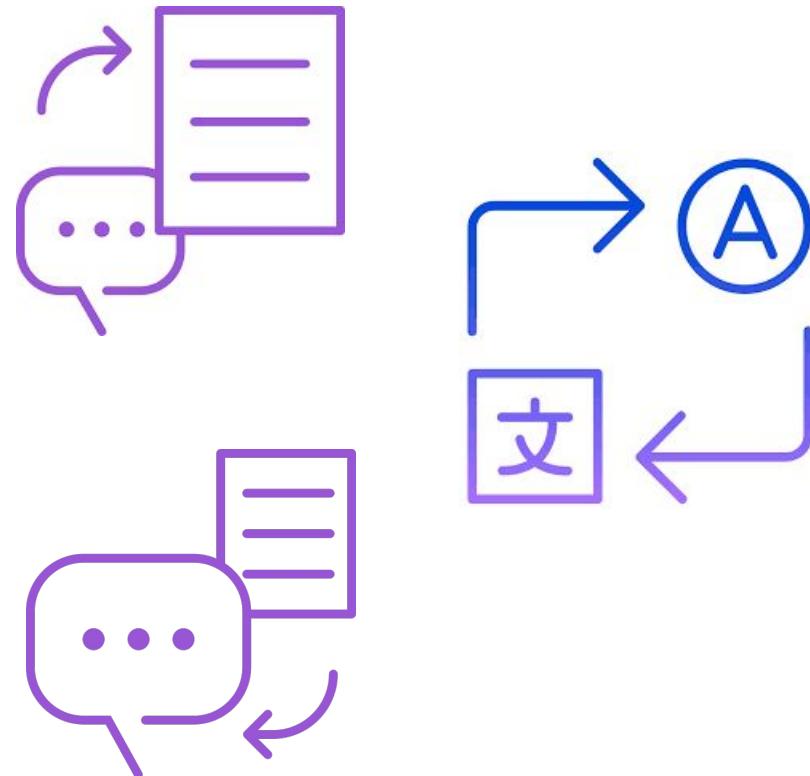
**Reto 4:** En el archivo *Retos\_Clase5\_Watson* y en 10 min, complete las celdas restantes del archivo para que el parlante en inglés pueda contestar al parlante en español

**Tips:**

- Consulte en “San Google”
- Revisa las celdas de los retos anteriores.

**Resultado esperado:**

Audio con la traducción en español



Solución:

```
[ ] contes_ing = 'persona2_ingles.wav'
model_id = 'en-US_BroadbandModel'

input('Presiona enter para grabar')
grabar_audio(contes_ing)

[ ] # objeto para interactuar con Watson Speech to text
stt = SpeechToTextV1(iam_apikey=keys_watson.speech_to_text_key)

# se abre el archivo de audio
with open(contes_ing, 'rb') as audio_file:
    # Watson realiza la transcripción del audio
    result = stt.recognize(audio=audio_file,
                           content_type='audio/wav', model=model_id).get_result()

# se obtiene la transcripción del audio
text_eng = result['results'][0]['alternatives'][0]['transcript']
print(text_eng)
```

Solución:

```
[ ] # objeto para interactuar con Watson Language Translator
traductor = LanguageTranslatorV3(version='2018-05-31',
                                    iam_apikey=keys_watson.translate_key)

# se realiza la traducción
texto_traducido = traductor.translate(
    text = text_eng, model_id='en-es').get_result()

# se obtiene la lista de traducciones. Si el texto a traducir, tiene
# varias cadenas, la lista tendrá multiples entradas. En este caso
# se pasa una sola cadena, por lo que se tiene solo un elemento.
lista_traducciones = texto_traducido['translations'][0]

# se obtiene el texto de la traducción
traduccion_ingles = lista_traducciones['translation']

print(traduccion_ingles)
```

Solución:

```
[ ] contes_español = 'persona2_español.wav'

# objeto para interactuar con Watson Text to Speech
# create Text to Speech client
tts = TextToSpeechV1(iam_apikey=keys_watson.text_to_speech_key)

# abre el documento y graba una voz sintetizada en un archivo
with open(contes_español, 'wb') as audio_file:
    audio_file.write(tts.synthesize(traduccion_ingles,
                                    accept='audio/wav', voice='es-ES_EnriqueV3Voice').get_result().content)

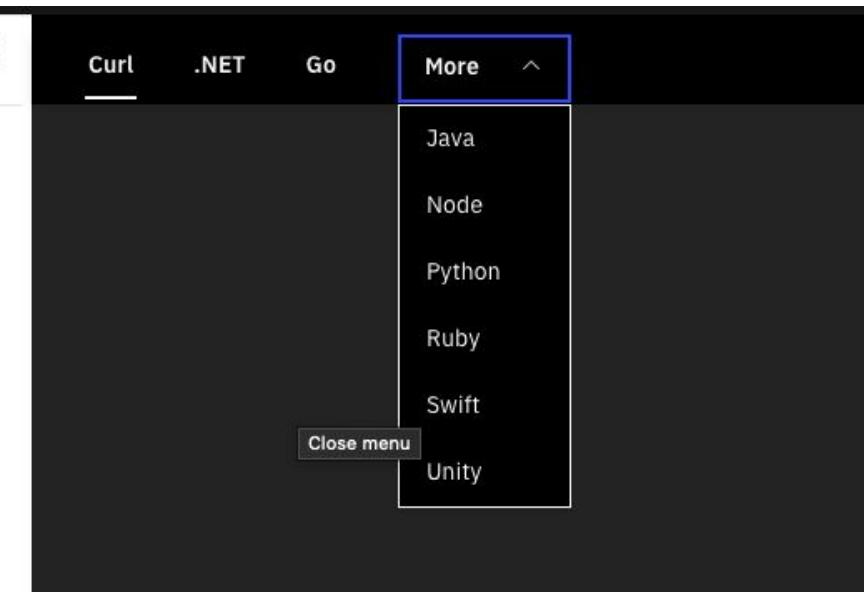
[ ] # se utiliza el módulo pydub para reproducir el archivo WAV
sound = pydub.AudioSegment.from_wav(contes_español)
pydub.playback.play(sound)
```

## Recursos de Watson

IBM nos brinda una amplia gama de recursos para los desarrolladores para que se familiaricen con los servicios y que los utilicen para sus aplicaciones.

<https://cloud.ibm.com/developer/watson/documentation>

En la documentación de la API de cada servicio muestra los detalles para interactuar con dicho servicio utilizando diversos lenguajes en los que está incluido Python. De igual forma que trabajamos con un SDK para Python, existen para otros lenguajes de programación



## Costo de los servicios de Watson

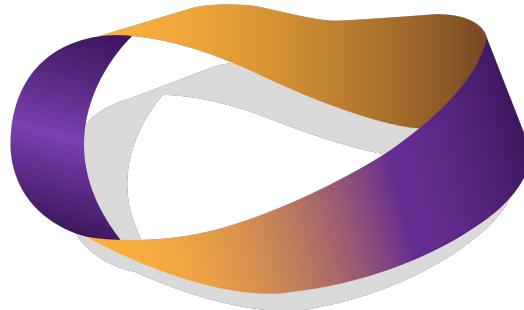
Los servicios de Watson permiten una versión gratuita (Lite), pero limitada. Para poder tener acceso a mayores características de cada uno de los servicios, existen cuotas que se deben de pagar. Por ejemplo el servicio de traducción tiene las siguientes tarifas

Plan	Características	Tarifas
Lite	Traduzca hasta 1.000.000 de caracteres al mes Identifique hasta 68 idiomas con la Identificación de idioma La traducción de documentos admite hasta 12 tipos de archivo	Gratis
Estándar	Todo lo incluido en el plan Lite, y además... Eliminación del límite de traducción de 1 millón de caracteres Los primeros 250.000 caracteres son gratuitos	0,02 \$ USD/THOUSAND CHAR
Advanced	Traducciones estándar Traducciones personalizadas Genere modelos personalizados de dominios específicos (prorrateados a diario)	0,02 \$ USD/THOUSAND CHAR 0,08 \$ USD/THOUSAND CHAR 15,00 \$ USD/INSTANCE MONTH

Consulte la documentación de cada servicio para conocer sus tarifas.

# Aprendizaje automático

para clasificación, regresión y agrupamiento de datos

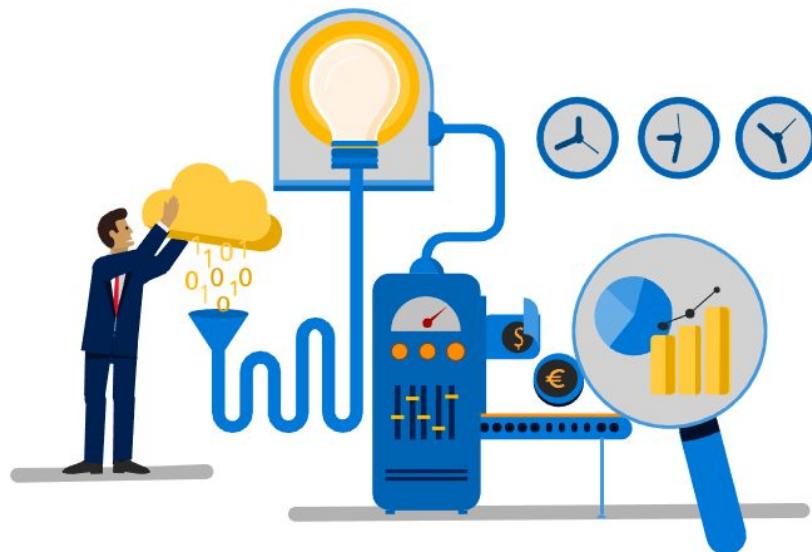


**ACTUMLOGOS**

DESARROLLANDO HABILIDADES TECNOLÓGICAS

## Introducción al aprendizaje automático

El aprendizaje automático es un campo dentro de la inteligencia artificial que ha tenido un gran impacto en los últimos años. Podremos resolver problemas que, hasta hace unos años, no se había podido resolver con un grado aceptable de eficacia. El aprendizaje automático es un gran campo de conocimiento por lo que vamos dar una introducción a las principales técnicas.



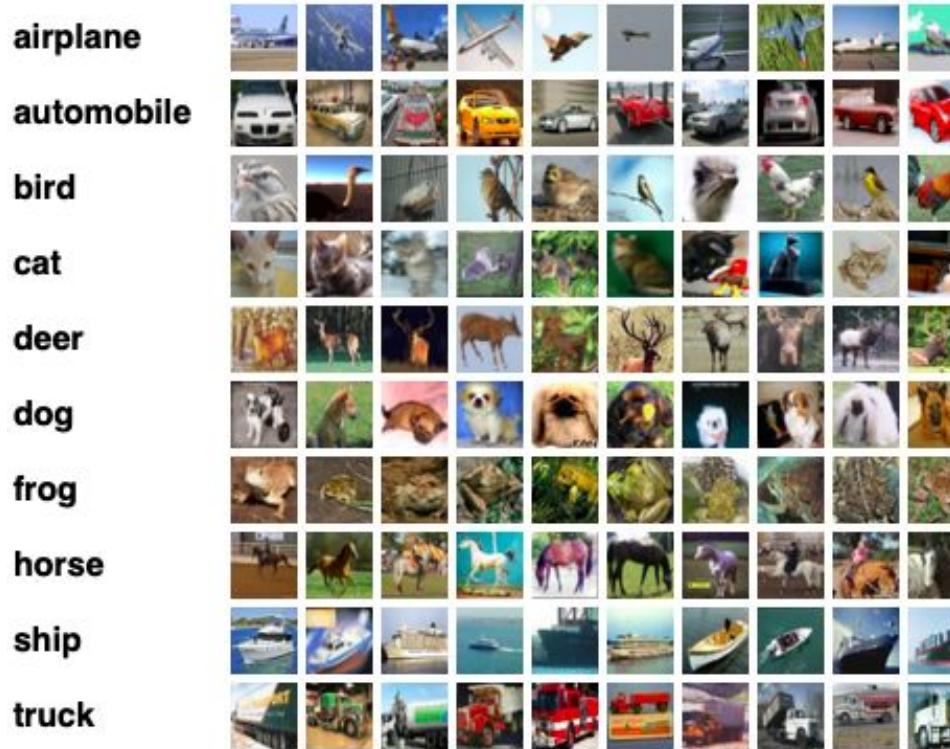
## ¿Podemos lograr que las máquinas aprendan?

La clave detrás del aprendizaje automático está en los datos. Anteriormente, la inteligencia artificial buscaba la forma de como programar la experiencia en los programas, actualmente la IA centra sus esfuerzos en hacer el programa aprenda de los datos: adquiera experiencia automáticamente.



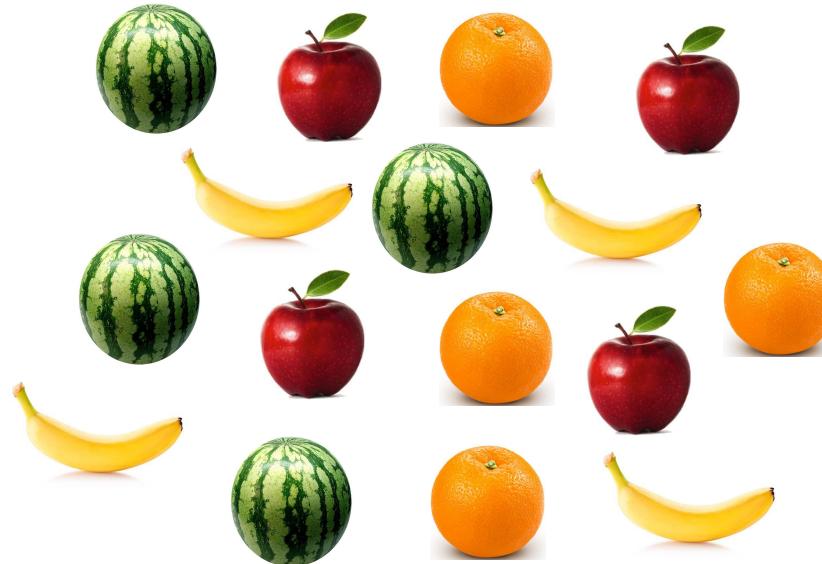
## Tipos de aprendizaje automático

- Aprendizaje **supervisado**: Para poder realizarse se trabaja con datos etiquetados, es decir, cada dato está asignado a una *clase* o etiqueta.



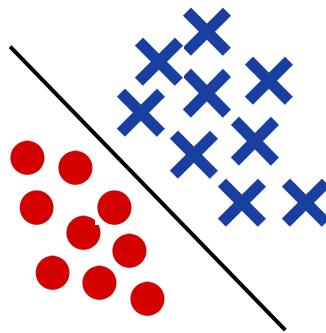
[The CIFAR-10 dataset](#)

- Aprendizaje **no supervisado**: los datos con los que se trabaja no están asociados a ninguna etiqueta, es decir que se dispone de datos pero se desconoce a la categoría a la cual pertenecen. El problema consiste en agrupar los datos similares ú obtener una representación de los datos más distinguible y compacta.

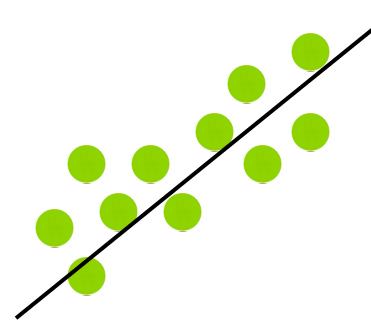


El aprendizaje supervisado se emplea para dos tipo de tareas:

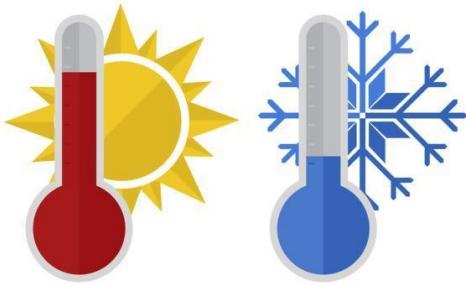
- **Clasificación:** este tipo de algoritmos predicen la categoría discreta a la cual pertenece un dato dado. Generalmente, se trabaja con problemas de tipo binarios o multiclas.
- **Regresión:** la predicción es una salida continua. Estos algoritmos generan un modelo que se aproxima a los datos para así poder realizar la predicción.



Clasificación



Regresión



**¿Hará frío o calor mañana?**



Problema de clasificación



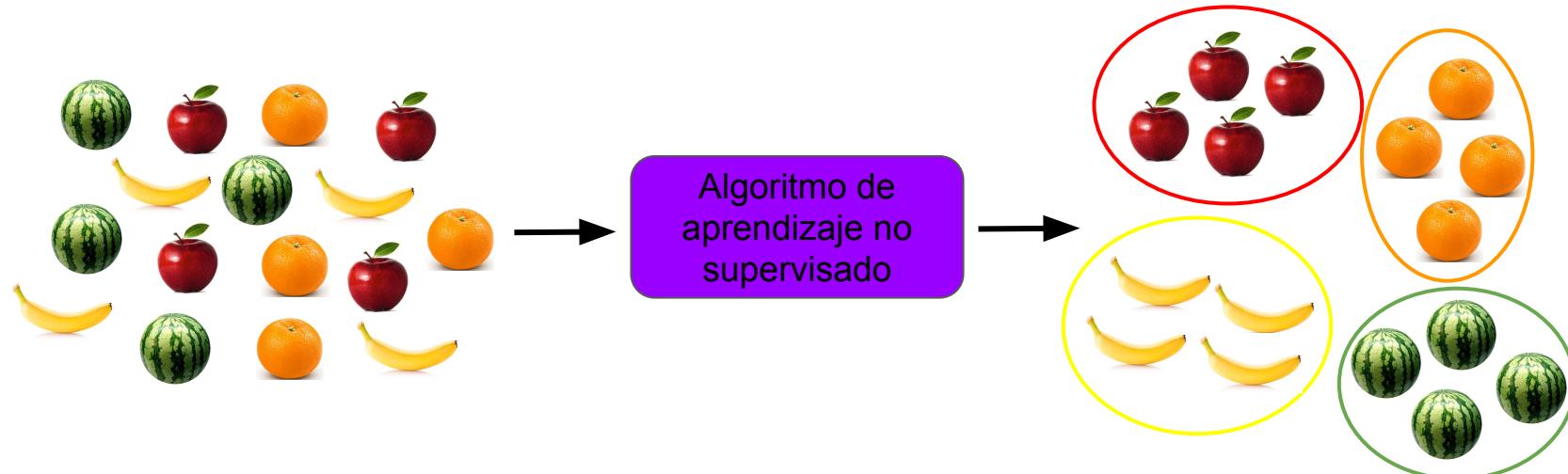
**¿Qué temperatura hará mañana?**



Problema de regresión

Por otra parte, el aprendizaje no supervisado se divide en

- **Agrupamiento (clustering):** busca reconocer similaridades en los datos, de tal forma que se puedan agrupar.
- **Reducción de dimensionalidad:** busca encontrar características que son relevantes, y así, poder reducir el costo computacional de los algoritmos supervisados.



## **Pasos en un estudio típico de ciencia de datos**

1. Cargar el conjunto de datos.
2. Explorar los datos.
3. Transformación de los datos (convertir datos no numéricos en numerosicos).
4. Dividir los datos en entrenamiento y prueba.
5. Creación del modelo.
6. Entrenamiento y prueba del modelo.
7. Ajuste del modelo y evaluación de su desempeño.
8. Hacer predicción con datos no vistos por el modelo.

## Caso de estudio 1: clasificación de dígitos con el algoritmo de *k* vecinos más cercanos

El servicio postal está en búsqueda de un sistema de inteligencia artificial que permita agilizar el proceso de poner en ruta el correo. Se requiere que dicho sistema sea capaz de reconocer los dígitos del código postal, los cuales están escritos a mano.



Lo primero que debemos hacer es identificar de qué tipo de problema se trata:

1. Es un problema de *clasificación multiclas*, se tienen 10 dígitos diferentes (del 0 al 9), es decir **hay 10 categorías o clases distintas**.
2. Se cuenta con un *dataset* compuesto por 1797 imágenes de dígitos escritos a mano de 8x8 pixeles. Este *dataset* fue creado a partir de otro llamado [MNIST](#).
3. El dataset está etiquetado, es un problema de **aprendizaje supervisado**.



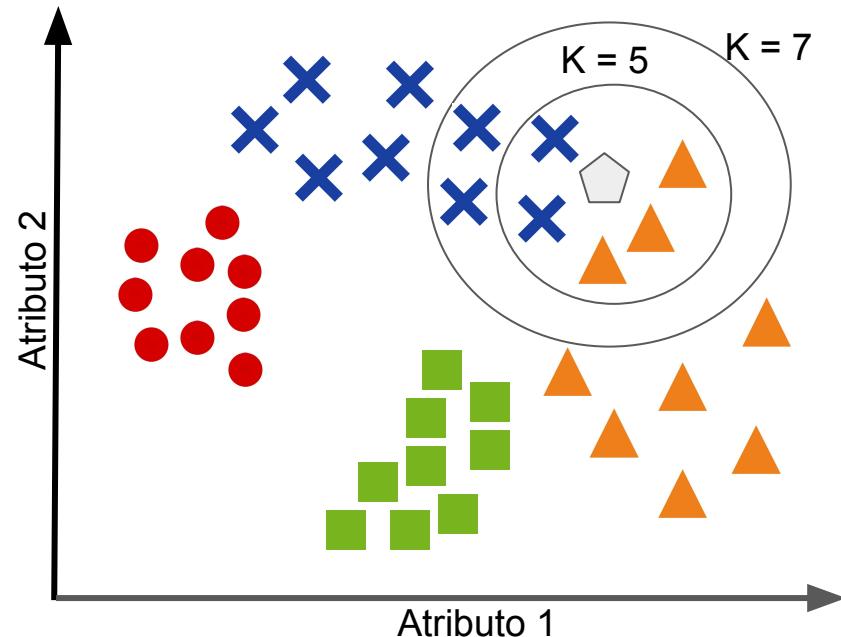
En este punto es importante seleccionar un *modelo*, es decir, una representación matemática de un proceso del mundo real que para una entrada determinada, produce una salida o predicción. Para generar un modelo se necesita proveer datos de entrenamiento a un algoritmo. Existen muchos modelos de aprendizaje automático que se utilizan para clasificación, uno de ellos es el llamado *k vecinos más cercanos (k-nearest neighbors)*.

Este algoritmo busca los  $k$  datos de entrenamiento más cercanos (en distancia) al dato de prueba. Los datos están descritos por  $p$  atributos o rasgos, de tal forma que el  $i$ -ésimo dato:

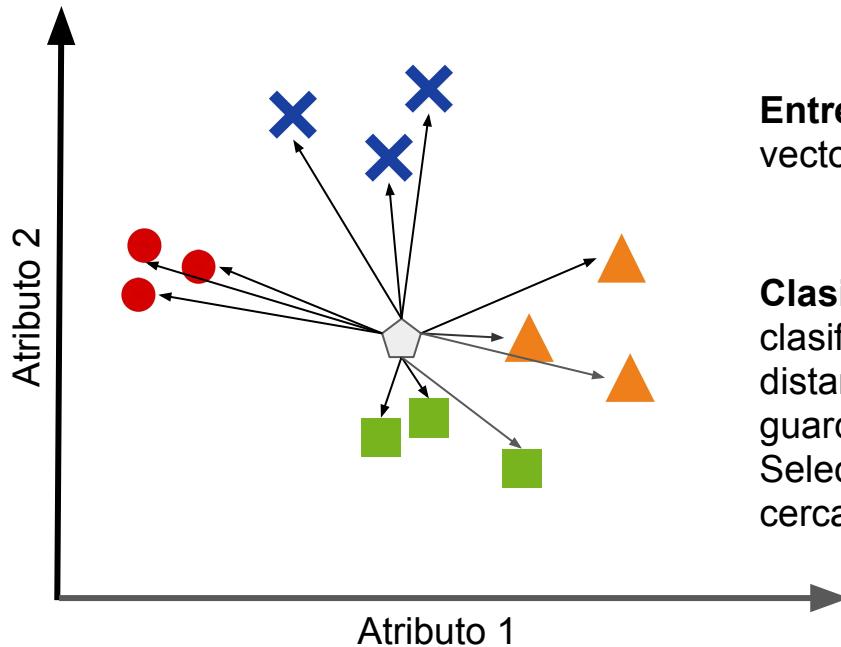
$$x_i = (x_{1i}, x_{2i}, \dots, x_{pi}) \in X$$

Considerando la distancia euclídea entre dos datos:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^p (x_{ri} - x_{rj})^2}$$



El **entrenamiento** de este algoritmo consiste en almacenar los vectores característicos de los datos de entrenamiento (su ubicación en el espacio *n-dimensional*) con sus respectivas etiquetas (clases). En la **clasificación**, se le presenta al modelo un nuevo dato del cual no se conoce su clase y se calcula su distancia con respecto a los vectores almacenados en el entrenamiento. Posteriormente, se seleccionan los  $k$  datos más cercanos y el nuevo dato es asignado a la clase que se repite más en los  $k$  vectores seleccionados.



**Entrenamiento:** Guardar los vectores característicos

**Clasificación:** Dado un dato a clasificar, obtener todas las distancias a los datos guardados en el entrenamiento. Seleccionar los  $k$  vecinos más cercanos

## Hiperparámetros (Hyperparameters)

Los modelos (algoritmos de aprendizaje automático) tienen dos tipos de parámetros

- **Parámetros de aprendizaje.** Aquellos que se calculan al momento de *aprender* de los datos que se le proveen.
- **Hiperparámetros.** Aquellos que el científico de datos especifique al momento de crear el modelo previo al entrenamiento.

En el caso del algoritmo de *k vecinos más cercanos*, *k* es un hiperparámetro por lo que debemos experimentar con valores diferentes de *k* para producir el mejor modelo posible para el caso de estudio. Este proceso se le conoce como **ajuste de hiperparámetros**.



## Scikit-Learn

Esta es un biblioteca muy popular de aprendizaje automático, incluye gran cantidad de algoritmos. Gracias a esta biblioteca es posible crear poderosos modelos de aprendizaje automático en unas cuantas líneas de código Python.

Instalación:

```
conda install scikit-learn
```

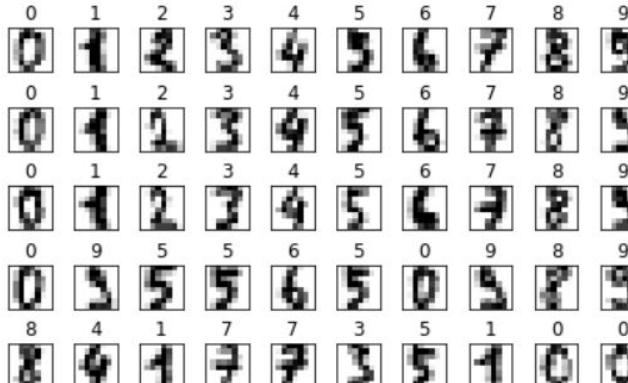


**Reto 5:** En el archivo *Retos\_Clase5\_ML* y en 5 min, completa las celdas del reto. Carga el dataset, explora los datos y gráfica algunos de ellos con sus respectivas etiquetas.

### Resultado esperado:

Las dimensiones de las imágenes son (1797, 8, 8)

```
[[ 0.  0.  5. 13.  9.  1.  0.  0.]  
 [ 0.  0. 13. 15. 10. 15.  5.  0.]  
 [ 0.  3. 15.  2.  0. 11.  8.  0.]  
 [ 0.  4. 12.  0.  0.  8.  8.  0.]  
 [ 0.  5.  8.  0.  0.  9.  8.  0.]  
 [ 0.  4. 11.  0.  1. 12.  7.  0.]  
 [ 0.  2. 14.  5. 10. 12.  0.  0.]  
 [ 0.  0.  6. 13. 10.  0.  0.  0.]]  
 [ 0.  0.  5. 13.  9.  1.  0.  0.  0. 13. 15. 10. 15.  5.  0.  0.  3.  
 15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.  
 0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.  
 0.  0.  0.  0.  6. 13. 10.  0.  0.  0.]]
```



### Tips:

- Consulte en “San Google”
- Importa el método *load\_digits()* de la biblioteca *sklearn.datasets*
- Importa el método *load\_digits()* de la biblioteca *sklearn.datasets*

Solución:

```
[ ] from sklearn.datasets import load_digits  
import matplotlib.pyplot as plt  
  
[ ] digits = load_digits()  
  
[ ] print(f'Las dimensiones de las imágenes son {digits.images.shape}')  
print(digits.images[0]) # se muestra los datos (matriz) de la primer imagen  
print(digits.data[0]) # se muestra los datos (vector) de la primer imagen  
  
[ ] figure, axes = plt.subplots(nrows=5, ncols=10, figsize=(6, 4))  
for item in zip(axes.ravel(), digits.images, digits.target):  
    axes, image, target = item  
    axes.imshow(image, cmap=plt.cm.gray_r)  
    axes.set_xticks([])  
    axes.set_yticks([])  
    axes.set_title(target)  
plt.tight_layout()
```

**Reto 6:** En el archivo *Retos\_Clase5\_ML* y en 10 min, separa los datos en un conjunto de entrenamiento y otro de prueba en una relación 80-20. Crea un modelo de *k vecinos más cercanos* y entrenalo. Por último prueba el modelo y calcula su error.

**Tips:**

- Consulte en “San Google”
- Importa el método `train_test_split()` de la biblioteca `sklearn.model_selection`
- Para crear el modelo use la clase `KNeighborsClassifier()`
- Entrene el modelo con el método `fit()`

**Resultado esperado:**

Datos de entrenamiento (1437, 64)

Etiquetas de entrenamiento (1437,)

Datos de prueba (360, 64)

Etiquetas de prueba (1437,)

La predicción es [2 1 5 6 9 9 5 6 4 4 1 8 0 9 9 6 1 9 0 4 5 1 6 3 1 1 1 0 3 6]

Lo esperado es [2 1 5 6 9 9 5 6 4 4 1 8 0 9 9 6 8 9 0 4 5 1 6 3 1 1 1 0 3 6]

## Solución:

```
[ ] from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    digits.data, digits.target, test_size=0.20)

[ ] print(f'Datos de entrenamiento {X_train.shape}')
print(f'Etiquetas de entrenamiento {y_train.shape}')
print(f'Datos de prueba {X_test.shape}')
print(f'Etiquetas de prueba {y_test.shape}')

[ ] from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5) # modelo clasificador
knn.fit(X=X_train, y=y_train) # entrenamiento del clasificador

[ ] predicted = knn.predict(X=X_test)
expected = y_test

# predicción sobre 30 elementos
print(f'La predicción es {predicted[:30]}')
print(f'Lo esperado es {expected[:30]}')

[ ] error = 0
for (p, e) in zip(predicted, expected):
    if p != e:
        error += 1
print(f'El error en porcentaje es de {(error/len(predicted)*100):.3f}')
```

## **¿Cuáles son la fortaleza del algoritmo de $k$ vecinos más cercanos?**

- Es un algoritmo simple de implementar.
- Es no paramétrico.

## **¿Cuáles son las debilidades del algoritmo de $k$ vecinos más cercanos?**

- Si hay muchos datos de entrenamiento, se vuelve costoso, computacionalmente hablando, ya que debe guardar todos los datos que se den para el entrenamiento y así poder calcular las distancias.
- Es sensible al ruido en los datos.
- Se ve afectado por la alta dimensionalidad.

## Caso de estudio 2: Series de tiempo y regresión lineal

En la vida real, es común encontrar colecciones de datos que representan variables independientes y dependientes. Por ejemplo, el registro de las temperaturas promedio de una ciudad a lo largo de un año. La temperatura es una variable dependiente y el año es la independiente. Un modelo de regresión lineal busca describir la relación que existe entre esas variables mediante una línea recta, que es conocida como *regresión lineal*.

Como conjunto de datos utilizaremos los datos de la temperatura promedio anual de la ciudad de Nueva York desde 1895 hasta 2018.



En un regresor lineal lo que se busca es que su salida (predicción) sea una combinación lineal de las características de los datos. Matemáticamente hablando sea  $\hat{y}$  la predicción

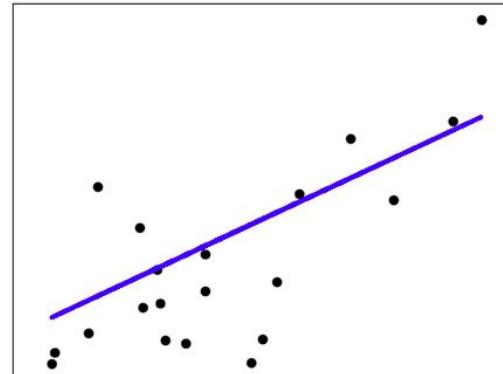
$$\hat{y}(w, x) = w_0 + w_1x_1 + \dots + w_px_p$$

donde  $w = (w_1, \dots, w_p)$  coeficientes para la combinación lineal y  $w_0$  es el umbral.

En la implementación de la biblioteca *scikit-learn* estos valores se obtienen en los atributos *coef\_* e *intercept\_* respectivamente.

El algoritmo busca minimizar la suma residual del error cuadrático entre las etiquetas predichas y las que viene con los datos

$$\min_w ||Xw - y||_2^2$$



**Reto 7:** En el archivo *Retos\_Clase5\_ML* y en 10 min, carga los datos que está en el archivo CSV, explora los datos, separarlos para entrenamiento y prueba. Por último, realiza el entrenamiento del modelo para obtener sus parámetros

### Resultado esperado:

	Fecha	Temperatura
0	1895	34.2
1	1896	34.7
2	1897	35.5
3	1898	39.6
4	1899	36.4
5	1900	37.4
6	1901	37.0
7	1902	35.0
8	1903	35.5
9	1904	29.8

Dimensiones de los datos de entrenamiento (93, 1)

Dimensiones de los datos de prueba (31, 1)

Los parámetros del modelo son  
[0.01652625] y [0.01652625]

### Tips:

- Consulte en “San Google”
- Importa la clase `LinearRegression` de la biblioteca `sklearn.linear_model`
- Importa las bibliotecas `pandas`, `numpy` y `seaborn`
- Importa el método `train_test_split()` de la biblioteca `sklearn.model_selection`
- Para crear el modelo use la clase `LinearRegression()`
- Entrene el modelo con el método `fit()`

## Solución:

```
[1] # Se importan bibliotecas
    import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.linear_model import LinearRegression

[2] # Cargando los datos de temperatura con Pandas
    nyc = pd.read_csv('nyc_temp_1895-2018.csv')

    nyc.columns = ['Fecha', 'Temperatura']

    # Explorando y ajustando los datos
    nyc.Fecha = nyc.Fecha.floordiv(100)

    nyc.head(10)

[3] # Separando los datos en entrenamiento y prueba
    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(
        nyc.Fecha.values.reshape(-1, 1), nyc.Temperatura.values)

    print(f'Dimensiones de los datos de entrenamiento {X_train.shape}')
    print(f'Dimensiones de los datos de prueba {X_test.shape}')

[4] # Creación y entrenamiento del modelo
    linear_regression = LinearRegression()

    linear_regression.fit(X=X_train, y=y_train)

    print('Los parámetros del modelo son')
    print(f'{linear_regression.coef_} y {linear_regression.intercept_}'')
```

**Reto 8:** En el archivo *Retos\_Clase5\_ML* y en 5 min, prueba el modelo entrenado en el reto anterior con el conjunto de prueba. Realiza la predicción de la temperatura para el año 2020 y por último gráfica los datos del *dataset* junto con la regresión lineal.

### Resultado esperado:

Predicción: 38.01, Real: 42.40

Predicción: 38.44, Real: 36.10

Predicción: 38.04, Real: 39.60

Predicción: 37.50, Real: 36.40

Predicción: 37.30, Real: 40.20

Predicción: 38.03, Real: 46.00

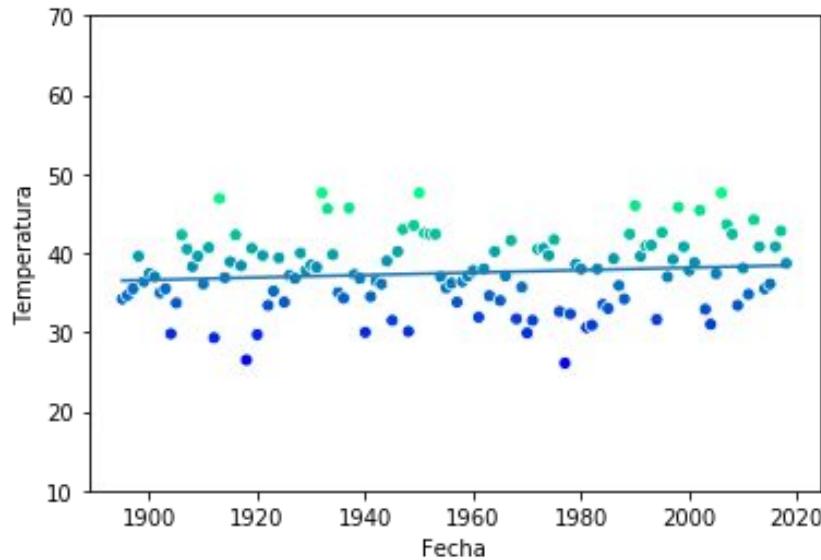
Predicción: 37.94, Real: 33.00

Año de la predicción 2020

La temp del año 2020 es 38.52

### Tips:

- Consulte en “San Google”
- Utiliza el método `predict()` del objeto de regresión.
- Emplea los atributos del modelo `.coef_` e `.intercept_` para hacer predicciones



Solución:

```
[5] # Probando el modelo
pred = linear_regression.predict(X_test)
real = y_test

for p, e in zip(pred[::5], real[::5]):
    print(f'Predicción: {p:.2f}, Real: {e:.2f}')

año = np.array(int(input('Año de la predicción ')))
p = linear_regression.predict(año.reshape(1,1))
print(f'La temp del año {año} es {p[0]:.2f}')

[6] # Se utilizan los parámetros ajustados en el entrenamiento
predict = (lambda x: linear_regression.coef_*x +
            linear_regression.intercept_)

print(predict(2020))
print(predict(1890))

[7] # Visualizando el conjunto de datos y la linea de regresión
axes = sns.scatterplot(data=nyc, x='Fecha', y='Temperatura',
                        hue='Temperatura', palette='winter', legend=False)

axes.set_ylim(10, 70)
x = np.array([min(nyc.Fecha.values), max(nyc.Fecha.values)])
y = predict(x)
p = plt.plot(x, y)
```

## Caso de estudio 3: regresión lineal multiple

En el caso anterior teníamos un *dataset* relativamente sencillo, sin embargo, en el mundo real los conjuntos de datos suelen ser de una complejidad mayor tanto en cantidad de datos como en sus características. Ahora vamos a ocupar el *California Housing dataset* el cual está compuesto con 20640 datos cada uno de ellos con 8 características numéricas.



**Reto 9:** En el archivo *Retos\_Clase5\_ML* y en 10 min, completa las celdas. Carga el dataset, imprime su descripción, explora los primeros 3 datos del dataset y añade las etiquetas como una columna del dataset.

## Resultado esperado:

California Housing dataset

```
-----  
**Data Set Characteristics:**  
  
:Number of Instances: 20640  
  
:Number of Attributes: 8 numeric, predictive attributes and the target  
  
:Attribute Information:  
- MedInc median income in block  
- HouseAge median house age in block  
- AveRooms average number of rooms  
- AveBedrms average number of bedrooms  
- Population block population  
- AveOccup average house occupancy  
- Latitude house block latitude  
- Longitude house block longitude
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.9841	1.0238	322.0	2.5556	37.88	-122.23
1	8.3014	21.0	6.2381	0.9719	2401.0	2.1098	37.86	-122.22
2	7.2574	52.0	8.2881	1.0734	496.0	2.8023	37.85	-122.24

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseValue
0	8.3252	41.0	6.9841	1.0238	322.0	2.5556	37.88	-122.23	4.526
1	8.3014	21.0	6.2381	0.9719	2401.0	2.1098	37.86	-122.22	3.585
2	7.2574	52.0	8.2881	1.0734	496.0	2.8023	37.85	-122.24	3.521

## Tips:

- Consulte en “San Google”
- Importa el método `fetch_california_housing()` de la biblioteca `sklearn.datasets`
- Utiliza el atributo `DESCR` del dataset
- Crea un `DataFrame` de pandas con los datos
- Utiliza el método `Series()` del objeto de la biblioteca pandas

Solución:

```
[1] # Cargando bibliotecas
    from sklearn.datasets import fetch_california_housing
    import matplotlib.pyplot as plt
    import seaborn as sns
    import pandas as pd

[2] # Cargando el dataset
    california = fetch_california_housing()
    # Mostrando la descripción del dataset
    print(california.DESCR)

[3] print(california.data.shape)
    print(california.target.shape)

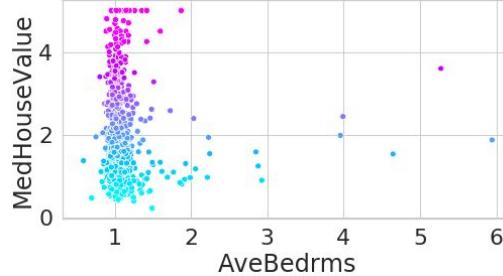
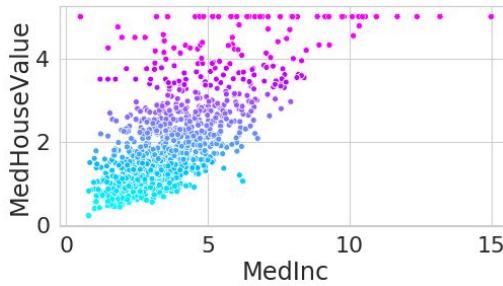
[4] # Explorando los datos
    pd.set_option('precision', 4)
    pd.set_option('display.width', None)

    california_df = pd.DataFrame(california.data,
                                    columns=california.feature_names)
    california_df.head(3)

[5] california_df['MedHouseValue'] = pd.Series(california.target)
    california_df.head()
```

**Reto 10:** En el archivo *Retos\_Clase5\_ML* y en 10 min, completa las celdas. Gráfica las características de los datos, tomando una muestra del 5% de los datos. Separa los datos para entrenamiento y prueba. Realiza el entrenamiento del modelo y prueba su funcionamiento con 5 datos.

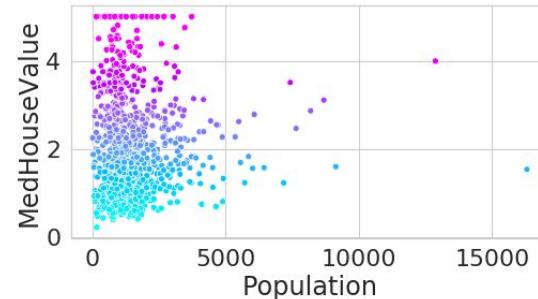
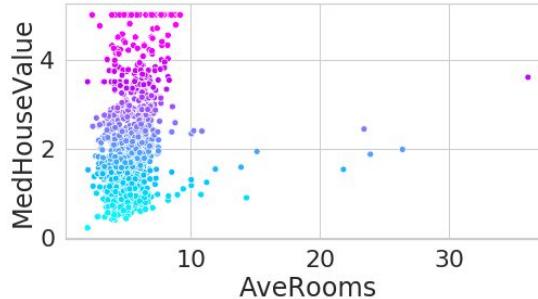
### Resultado esperado:



(15480, 8)

(5160, 8)

```
[1.18394736 1.75138558 2.26335662 1.34539847 2.15683322]  
[1.013 2.331 1.775 0.225 0.982]
```



### Tips:

- Consulte en “San Google”
- Utilice el método `sample()` para tomar la muestra
- Importa la clase `LinearRegression` de la biblioteca `sklearn.linear_model`
- Importa el método `train_test_split()` de la biblioteca `sklearn.model_selection`
- Para crear el modelo use la clase `LinearRegression()`
- Entrene el modelo con el método `fit()`

## Solución:

```
[6] # Observando las características de los datos
    sample_df = california_df.sample(frac=0.05)

    sns.set(font_scale=2)
    sns.set_style('whitegrid')

    for feature in california.feature_names:
        plt.figure(figsize=(8, 4))
        sns.scatterplot(data=sample_df, x=feature,
                        y='MedHouseValue', hue='MedHouseValue',
                        palette='cool', legend=False)

[7] # Separando los datos para entrenamiento y prueba
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    california.data, california.target)

print(X_train.shape)
print(X_test.shape)

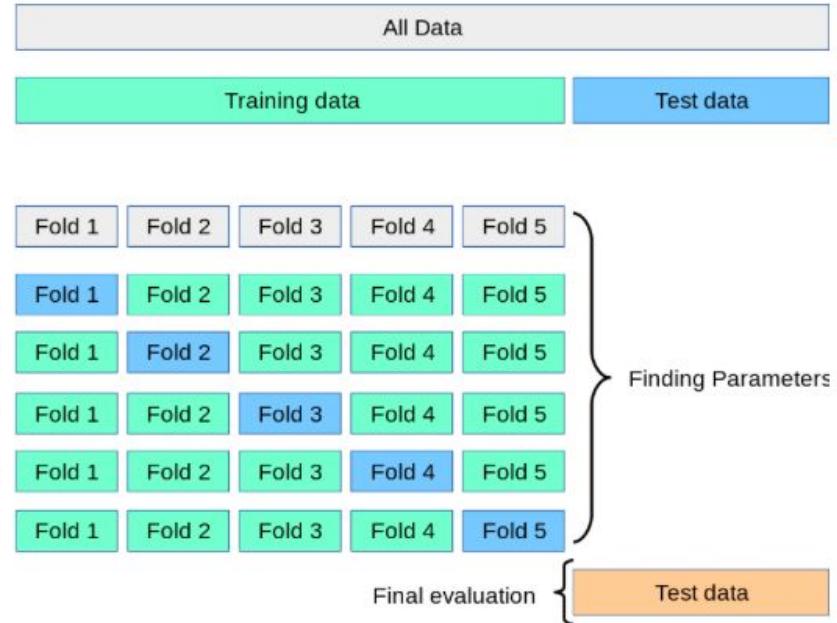
[8] # Entrenando el modelo
from sklearn.linear_model import LinearRegression

linear_regression = LinearRegression()
linear_regression.fit(X=X_train, y=y_train)

[9] # Probando el modelo
predicted = linear_regression.predict(X_test)
expected = y_test
print(predicted[:5])
print(expected[:5])
```

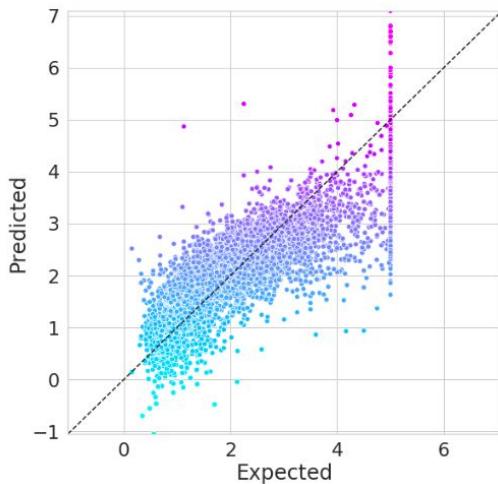
## Validación cruzada

Para obtener los mejores resultados en cuanto a la medición de la calidad de la predicción de los modelos, es mejor realizar una validación cruzada. Se divide el conjunto de entrenamiento en  $k$  subconjuntos y, al momento de realizar el entrenamiento, se va a tomar cada  $k$  subconjunto como conjunto de *prueba* del modelo. Este proceso se repetirá  $k$  veces, y en cada iteración se seleccionará un conjunto de *prueba* diferente. Una vez finalizadas las iteraciones, se calcula la precisión y el error para cada uno de los modelos producidos, y para obtener la precisión y el error final se calcula el promedio de los  $k$  modelos entrenados.



**Reto 11:** En el archivo *Retos\_Clase5\_ML* y en 5 min completa las celdas. Gráfica los resultados del valor esperado vs predicho por el modelo y muestra las métricas del modelo. Realiza la validación cruzada para elegir el mejor modelo.

### Resultado esperado:



0.6274296619175215  
0.49374569480529656

```
LinearRegression: mean of r2 scores=0.599
ElasticNet: mean of r2 scores=0.423
Lasso: mean of r2 scores=0.285
Ridge: mean of r2 scores=0.599
```

### Tips:

- Consulte en “San Google”
- Utiliza el método *Series()* del objeto de la biblioteca pandas
- Importa la clase *metrics* de la biblioteca de sklearn
- Utiliza los métodos *r2\_score()* y *mean\_squared\_error* de *metrics*
- Importa las clases *ElasticNet*, *Lasso* y *Ridge* de la biblioteca *sklearn.linear\_model*
- Importa el método *Kfold* y *cross\_val\_score* de la biblioteca *sklearn.model\_selection*

## Solución:

```
[10] # Visualizando los precios esperado y los predichos
df = pd.DataFrame()
df['Expected'] = pd.Series(expected)
df['Predicted'] = pd.Series(predicted)

figure = plt.figure(figsize=(9, 9))
axes = sns.scatterplot(data=df, x='Expected', y='Predicted',
hue='Predicted', palette='cool', legend=False)

start = min(expected.min(), predicted.min())
end = max(expected.max(), predicted.max())
axes.set_xlim(start, end)
axes.set_ylim(start, end)
line = plt.plot([start, end], [start, end], 'k--')

[11] # Metricas de la regresión
from sklearn import metrics

print(metrics.r2_score(expected, predicted))
print(metrics.mean_squared_error(expected, predicted))
```

```
[12] # Escogiendo el mejor modelo
from sklearn.linear_model import ElasticNet, Lasso, Ridge
estimators = {
    'LinearRegression': linear_regression,
    'ElasticNet': ElasticNet(),
    'Lasso': Lasso(),
    'Ridge': Ridge()
}

from sklearn.model_selection import KFold, cross_val_score

for estimator_name, estimator_object in estimators.items():
    kfold = KFold(n_splits=10, random_state=11, shuffle=True)
    scores = cross_val_score(estimator=estimator_object,
                             X=california.data, y=california.target, cv=kfold,
                             scoring='r2')
    print(f'{estimator_name}: ' +
          f'mean of r2 scores={scores.mean():.3f}')
```

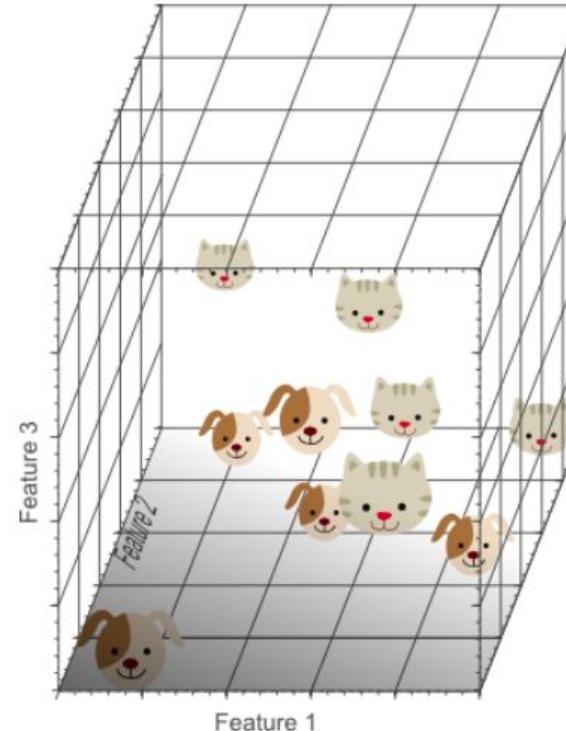
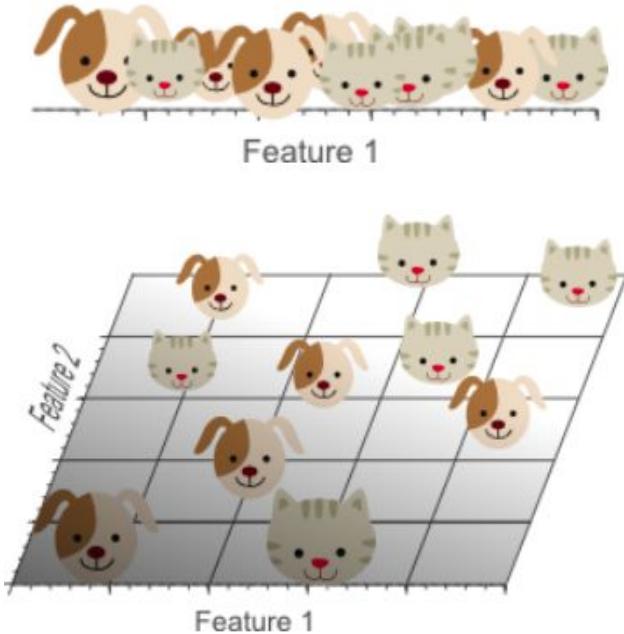
## Caso de estudio 4: aprendizaje no supervisado

### **Reducción de dimensionalidad**

El aprendizaje no supervisado y la visualización de datos permite encontrar *patrones* y relaciones entre muestras no etiquetadas. Cuando se trabaja con series de tiempo, la visualización es sencilla en una gráfica 2D, inclusive si se tuvieran tres dimensiones, la visualización sigue siendo posible.

Sin embargo, ¿cómo visualizar datos que poseen más dimensiones?





## ¿Reducir o aumentar la dimensionalidad?

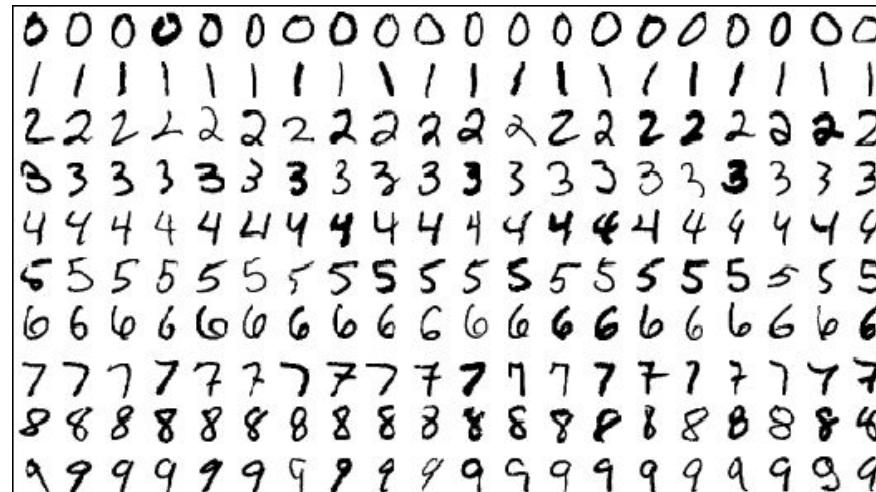
En casos reales, comúnmente se busca reducir la dimensionalidad.

Por ejemplo, el dataset de dígitos escritos a mano que trabajamos contiene datos con 64 características cada uno, sin embargo, existen otros conjuntos de datos que poseen cientos, miles o inclusive millones de características.

Para visualizar un dataset con tantas características, primero se reducen los datos a dos o tres dimensiones. Esto requiere utilizar la técnica de aprendizaje no supervisado llamada *reducción de dimensionalidad*. Esta técnica permite identificar patrones relevantes que están “ocultos” en la información, por ejemplo se pueden llegar a identificar agrupaciones que definen clases distintas y entonces utilizar algún algoritmo de clasificación (como el de *k vecinos más cercanos* que revisamos previamente).

Entrenar estimadores en conjuntos con un número significativo de dimensiones puede tomar, horas, días, semanas o inclusive mucho más. Esto se conoce como la *maldición de la dimensionalidad*. Si los datos poseen características con una alta correlación, entonces pueden ser eliminadas a través de la reducción de la dimensionalidad. Sin embargo, también pudiera reducir la precisión del modelo.

Utilizaremos el conjunto de los dígitos escritos a mano ignorando sus etiquetas para reducir su dimensionalidad a dos dimensiones.



El algoritmo que se va emplear para reducir la dimensionalidad se llama Incrustación estocástica de vecinos de distribución-T (t-distributed Stochastic Neighbor Embedding) o simplemente t-SNE. Dado un espacio de  $N$  dimensiones que contiene objetos  $\mathbf{x}_1, \dots, \mathbf{x}_N$  se calcula la probabilidad  $p_{ij}$  que es proporcional a la similaridad de dos objetos  $\mathbf{x}_i$  y  $\mathbf{x}_j$

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$

Se desea que la nueva dimensionalidad  $d$  tal que  $\mathbf{y}_i \in \mathbb{R}^d$  permita calcular una medida de similaridad entre dos puntos  $q_{ij}$

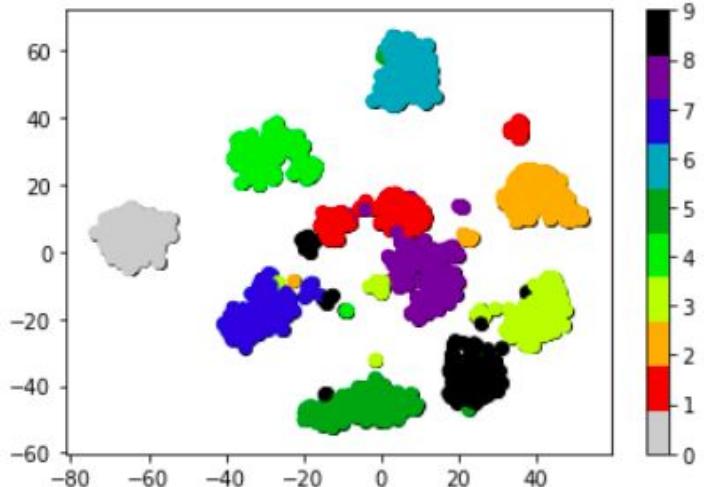
$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$

La localización de los puntos se obtiene al minimizar la función

$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

**Reto 12:** En el archivo *Retos\_Clase5\_ML* y en 5 min complete las celdas. Crea un estimador que permita reducir la dimensionalidad a dos, entrenalo con el dataset de dígitos escritos a mano y gráfica los resultados.

### Resultado esperado:



### Tips:

- Consulte en “San Google”
- Importa el método *load\_digits()* de la biblioteca *sklearn.datasets*
- Importa la clase TSNE de la biblioteca *sklearn.manifold*
- Implemente el estimador para dos dimensiones
- Utiliza el método *fit\_transform()* sobre los datos del conjunto de números

Solución:

```
[1] # Cargando bibliotecas
    from sklearn.datasets import load_digits
    from sklearn.manifold import TSNE
    import matplotlib.pyplot as plt

[2] # Cargando el dataset
    digits = load_digits()

    # Creando un estimador para la reducción de dimensionalidad
    tsne = TSNE(n_components=2)

[3] # Transformando las características del dataset a dos dimensiones
    reduced_data = tsne.fit_transform(digits.data)

    reduced_data.shape

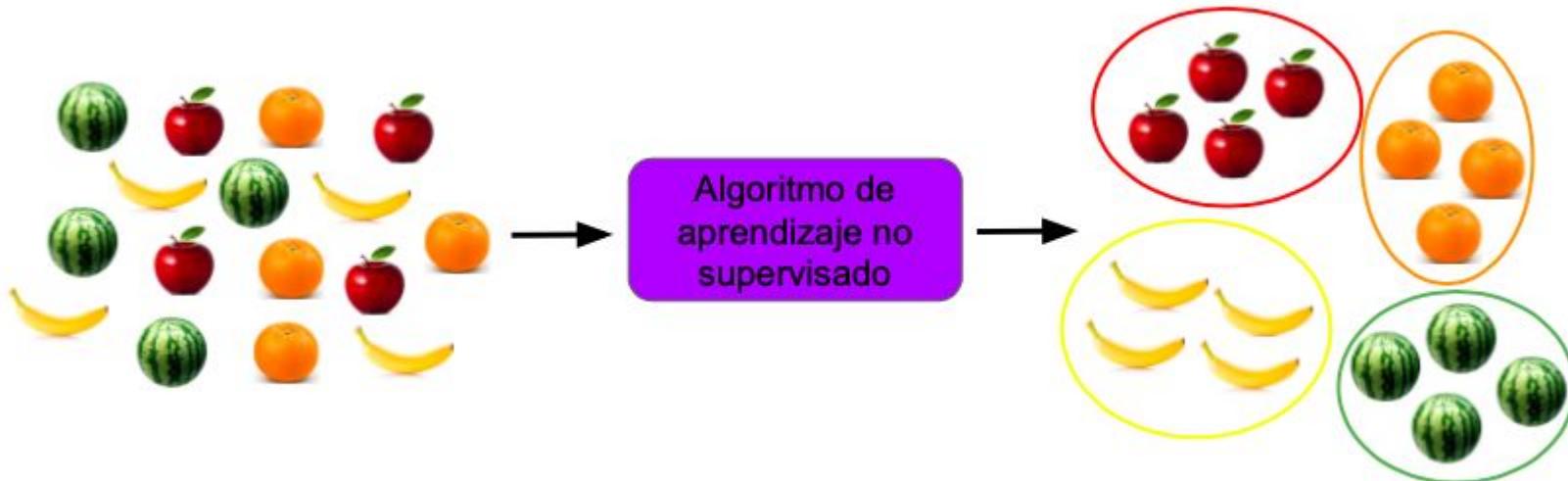
    # Visualizando los datos reducidos
    dots = plt.scatter(reduced_data[:, 0], reduced_data[:, 1],
                       c='black')

    # Visualizando los datos reducidos con diferentes colores para cada dígito
    dots = plt.scatter(reduced_data[:, 0], reduced_data[:, 1],
                       c=digits.target, cmap=plt.cm.get_cmap('nipy_spectral_r', 10))

    colorbar = plt.colorbar(dots)
```

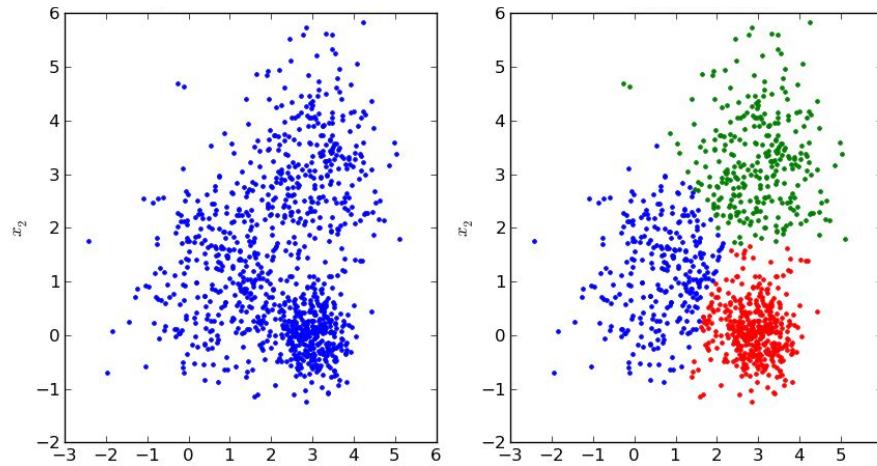
## Agrupamiento (*Clustering*)

Otras de las tareas comúnmente realizadas en el aprendizaje no supervisado es el *agrupamiento* (*clustering*). Lo que se desea es encontrar clases dentro de información que no está previamente clasificada (etiquetada). Uno de los algoritmos más utilizados para realizar esta tarea es el de *K-means*. La *K* representa el número de agrupamientos que se desea obtener. Como es de esperarse, este algoritmo realiza el agrupamiento usando el cálculo de distancias, como lo hace el algoritmo de *k vecinos más cercanos*.



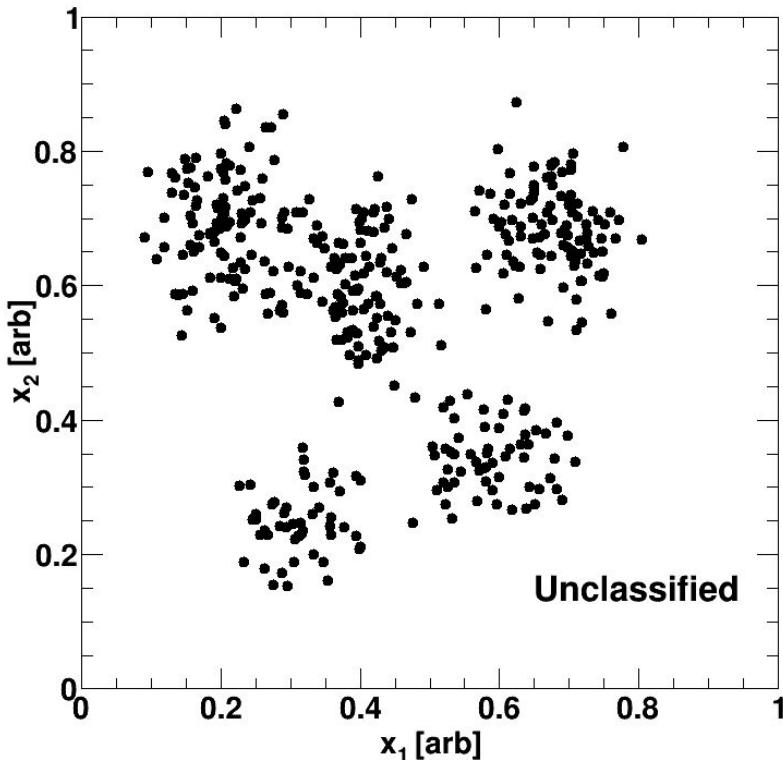
El algoritmo de *K-means* busca separar las muestras en  $n$  grupos con igual varianza. Este algoritmo recibe como entrada el número total de agrupamientos deseados. Lo primero que hace este algoritmo es dividir el conjunto en  $N$  muestras en  $K$  agrupamientos disjuntos  $C$ , donde cada uno está descrito por la media ( $\mu_j$ ) de sus muestras (centroides). El algoritmo ayuda a escoger los centroides que minimizan la inercia (medida de que tan unidos son los agrupamientos)

$$\sum_{i=0}^n \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$



## K-Means

Heurística: asume que el número de centros se conoce, y genera una hipótesis de solución que se ajusta iterativamente.



1. Generar  $k$  centros aleatoriamente  $\mu_1, \dots, \mu_k \in \mathbb{R}^n$
2. Repetir hasta  $N$  iteraciones

- a. Etiquetar cada dato en un grupo.

$$y_i := \arg \min_j \| \mathbf{x}_i - \mu_j \|^2$$

- b. Actualizar los centroides

$$\mu_j := \frac{\sum_{i=1}^m \mathbf{1}\{y_i = j\} \mathbf{x}_i}{\sum_{i=1}^m \mathbf{1}\{y_i = j\}}$$

Fin

$y_i$  Es la etiqueta que indica a qué grupo pertenece el dato  $i$ .

**Suboptimalidad:** Este algoritmo puede estancarse en un subóptimo local.

Vamos a trabajar con otro conjunto de datos muy conocido llamado *Iris Dataset*. Este conjunto agrupa 3 diferentes tipos de flor Iris: setosa, versicolor y virginica. Está etiquetado, sin embargo, vamos a ignorar estas etiquetas para demostrar el funcionamiento del algoritmo de agrupamiento. *Iris Dataset* está compuesto por 150 datos cada uno con cuatro características (longitud de sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo).



Iris setosa



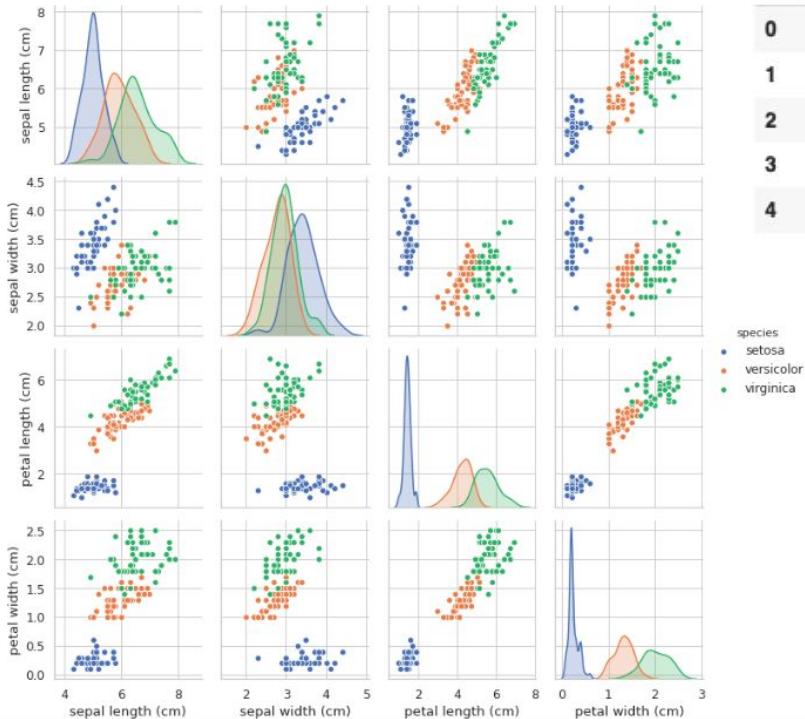
Iris versicolor



Iris virginica

**Reto 13:** En el archivo *Retos\_Clase5\_ML* y en 5 min complete las celdas. Carga el dataset *Iris*. Explora este dataset, muestra los primeros 5 elementos de este conjunto y gráfica sus características.

## Resultado esperado:



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

## Tips:

- Consulte en “San Google”
- Importa el método *load\_iris()* de la biblioteca *sklearn.datasets*
- Utiliza el método *head()* de la biblioteca *pandas*
- Importe la biblioteca *seaborn*
- Utiliza el método *pairplot()* de la biblioteca *seaborn*

## Solución:

```
[1] # Cargando el conjunto de datos
    from sklearn.datasets import load_iris

    iris = load_iris()

    # Checando el número de muestras, características y etiquetas
    print(iris.data.shape)
    print(iris.target.shape)
    print(iris.target_names)
    print(iris.feature_names)

[2] # Explorando el dataset
    import pandas as pd

    pd.set_option('max_columns', 5)
    pd.set_option('display.width', None)
    iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
    iris_df['species'] = [iris.target_names[i] for i in iris.target]

    iris_df.head()

[3] # Visualizando los datos con Seaborn
    import seaborn as sns

    sns.set(font_scale=1.1)

    sns.set_style('whitegrid')

    grid = sns.pairplot(data=iris_df, vars=iris_df.columns[0:4],
                        hue='species')
```

**Reto 14:** En el archivo *Retos\_Clase5\_ML* y en 5 min complete la celda. Cree un estimador basado en el algoritmo de K-means para el agrupamiento de datos y emplee dicho algoritmo para agrupar los datos del conjunto Iris

### Resultado esperado:

Agrupamiento para setosa

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

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

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

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

Agrupamiento para versicolor

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

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

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

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

Agrupamiento para virginica

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

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

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

### Tips:

- Consulte en “San Google”
- Importe la clase Kmeans de la biblioteca sklearn\_cluster
- Implemente el agrupamiento para 3 clases.
- Utilice el metodo fit() de la clase KMeans

Solución:

```
[4] # Agrupamiento (clustering) por K-medias

# Creando el estimador
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)

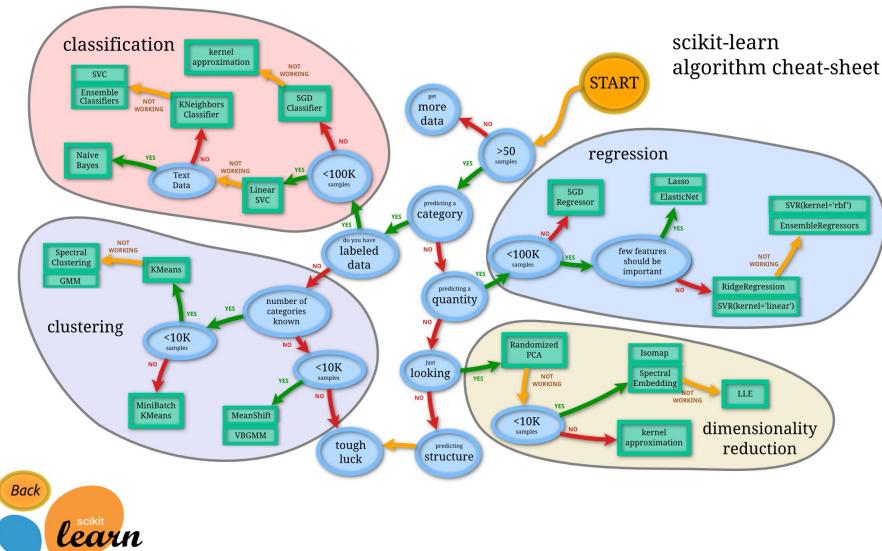
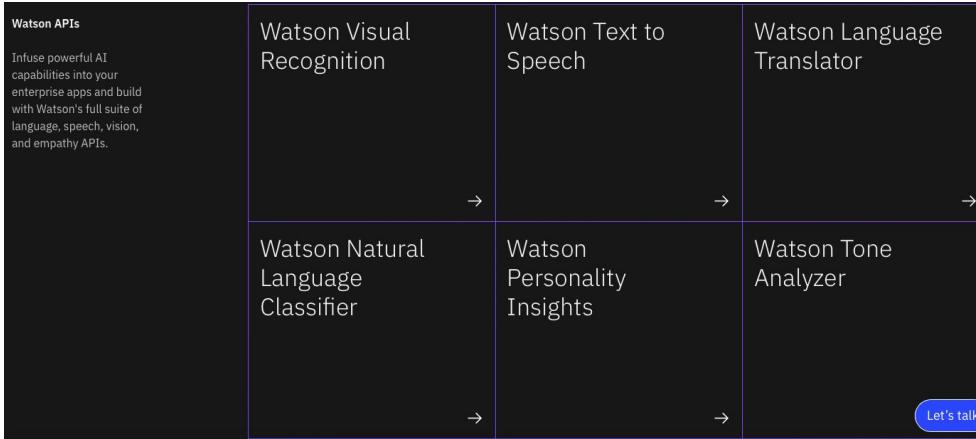
# Fitting the Model
kmeans.fit(iris.data)

# Resultados del agrupamiento
print('Agrupamiento para setosa')
print(kmeans.labels_[0:50])
print(iris.target[0:50])
print('Agrupamiento para versicolor')
print(kmeans.labels_[50:100])
print(iris.target[50:100])
print('Agrupamiento para virginica')
print(kmeans.labels_[100:150])
print(iris.target[100:150])
```

# iPlus ultra! ...ir más allá

Explora todos los servicios con los que cuenta Watson

## Servicios de Watson



Aprende más de aprendizaje automático con scikit-learn

scikit-learn

Guía de usuario

# Conclusión Final

- Watson es una potente tecnología cognitiva basada en la nube que permite resolver diversos problemas que van desde la traducción hasta la comprensión de documentos complejos. Cuenta con algoritmos de procesamiento de lenguaje natural para llevar a cabo gran parte de sus tareas.
- Los servicios de Watson cuentan con un plan *lite* el cual permite utilizarlos y ver sus principales funcionalidades.
- El aprendizaje automático es una área dentro de la inteligencia artificial que busca generar algoritmos que *aprendan* (modifiquen sus parámetros) a partir de datos.
- Existen dos tipos básicos de algoritmos de aprendizaje automático: supervisado y no supervisado.
- Dentro del aprendizaje supervisado se busca resolver problemas de clasificación y regresión.
- Dentro del aprendizaje no supervisado se busca resolver problemas de reducción de dimensionalidad y agrupamiento.

# Glosario

**Sistema cognitivo:** son aquellos sistemas que buscan resolver problemas de la forma en que lo haría un ser humano, entendiendo el lenguaje natural, analizando la información y dando resultados.

**Nube:** en el contexto de la computación, es una red mundial de servidores, dedicados a realizar funciones exclusivas.

**Aprendizaje automático:** subárea de la inteligencia artificial cuyo objetivo es desarrollar técnicas para que las computadoras *aprendan*.

**Clasificación:** tarea dentro del aprendizaje supervisado que consta en que un algoritmo sea capaz de asignar categorías a un conjunto de datos no categorizados.

**Regresión:** tarea dentro del aprendizaje supervisado que busca crear un modelo que se ajuste a los datos de entrenamiento para poder generar predicciones.

**Reducción de dimensionalidad:** tarea dentro del aprendizaje no supervisado que busca reducir la complejidad de los datos para su análisis.

**Agrupamiento:** tarea dentro del aprendizaje no supervisado que consiste en identificar grupos de elementos (clases) dentro de información que no está previamente clasificada.