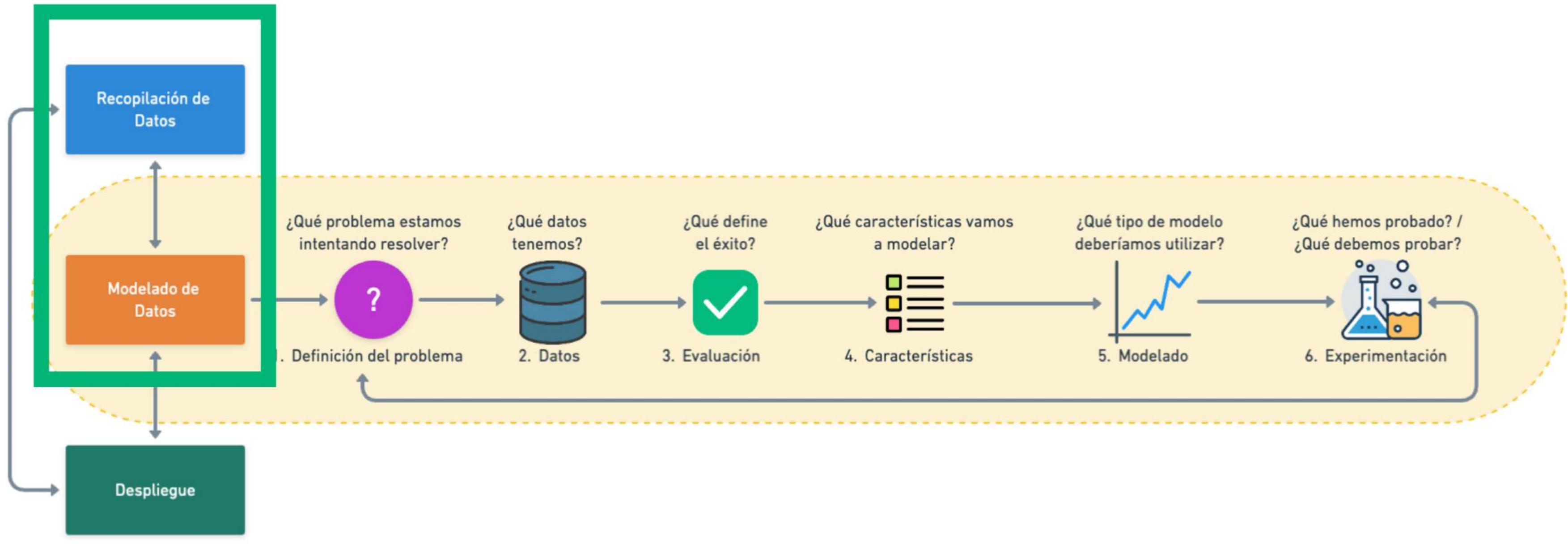


# MACHINE LEARNING

MTRO. ALFONSO GREGORIO RIVERO DUARTE

# FRAMEWORK





# **PREPROCESAMIENTO DE LOS DATOS**

# PREPROCESAMIENTO

1. Obtener el conjunto de Datos
2. Como importar librerías
3. Como importar Datasets
4. Datos faltantes
5. Datos categóricos
6. Cómo dividir el dataset en train y test
7. Como escalar los datos
8. Evaluación

# RETO

## Reto:

Comencemos con:

- Active su ambiente local creado con miniconda en su equipo o su código de COLAB
- Abra el archivo [05\\_Data\\_Preprocessing\\_Tools.ipynb](#)
- Ejecute el código y analícelo junto al instructor





## OBTENER EL CONJUNTO DE DATOS



Competitions Datasets Models Code Discussions Courses ...

Search

Sign In

Register

## Level up with the largest AI & ML community

Join over 18M+ machine learners to share, stress test, and stay up-to-date on all the latest ML techniques and technologies. Discover a huge repository of community-published models, data & code for your next project.

Register with Google

Register with Email



<https://www.kaggle.com/>

**GOOGLE**

## Dataset Search

Search for Datasets



Try [coronavirus covid-19](#) or [water quality site:canada.ca](#).

[Learn more](#) about Dataset Search.

<https://datasetsearch.research.google.com/>

On this page

Collections

## Awesome collections on DataHub

The awesome section presents collections of high quality datasets organized by topic.

Home page for awesome collections is located in the [frontend](#) repo and should be modified from there. See the live page here:

### Collections

- [Air Pollution data](#)
- [Bibliographic data](#)
- [Broadband data](#)

<https://datahub.io/collections>

# DATA WORLD

 data.world

Search data.world

Sign in

There are 131232 **free** datasets available on data.world.

Find open data about free contributed by thousands of users and organizations across the world.

TOP OPEN DATA TOPICS

- ↳ [economy](#) (1284)
- ↳ [hxl](#) (2103)
- ↳ [iatи](#) (1005)
- ↳ [education](#) (2832)
- ↳ [environment](#) (1706)
- ↳ [atmosphere](#) (2381)
- ↳ [geothermal](#) (1211)
- ↳ [oceans](#) (1271)
- ↳ [united states](#) (898)
- ↳ [active](#) (3268)
- ↳ [animal husbandry](#) (1055)
- ↳ [hospital](#) (1046)
- ↳ [funding](#) (900)
- ↳ [cso](#) (3142)
- ↳ [energy](#) (1052)

 **Exam Practice**  
Abby Hansen · Updated 3 years ago  
Dataset with 211 projects 1 file 3 tables  
412 Comment

 **Steven Seagal Box Office**  
Casey Jex Smith · Updated 7 years ago  
This dataset presents approximate figures for Steven Seagal's box office, and budget by film over time.  
Dataset with 534 projects 1 file 1 table  
Tagged stevenseagal  
898 Comment

How data.world helps you.

 Find fantastic data.  
Discover the data you need.  
Understand it at a glance.  
Follow people, topics, and projects.

[Discover data →](#)



<https://data.world/datasets/free>



# IMPORTAR LIBRERIAS EN PYTHON

# LIBRERIAS

Lo primero que debemos tener en cuenta es que, en Python, nosotros podemos importar objetos mediante las palabras reservadas `from` e `import`. Podremos importar de módulos y paquetes los objetos que deseemos, desde variables, constantes, pasando por funciones, clases hasta módulos y paquetes.

La primera opción, y la más sencilla, es usar el comando `import` seguido del nombre la librería. Una opción para no tener que llamar una librería por su nombre completo, es renombrarla al momento de importarla. Esto es una práctica bastante común, pues permite utilizar nombres cortos para las librerías, lo que hace mucho más fácil llamarlas.

**import math as mth**

# LIBRERIAS

Otra opción es sólo llamar un comando específico de una librería determinada. Para ello usamos la siguiente sintaxis:

**from math import sin**

En este caso sólo se carga el comando sin, sin cargar el resto de la librería. Esto puede ser muy ventajoso cuando la librería es grande y sólo queremos utilizar una pequeña funcionalidad de la misma. Otra ventaja es que no es necesario usar el nombre de la librería antes del comando.

La última opción consiste en llamar todos los comandos de una librería sin que estén precedidos por su nombre. Para ello se utiliza la sintaxis:

**from math import \***



# IMPORTAR DATASETS

# DATASETS

Para python:

```
import pandas as pd  
dataset = pd.read_csv("train.csv")
```

Para Collab:

```
from google.colab import drive  
drive.mount('/content/gdrive')  
dataset = pd.read_csv('gdrive/MyDrive/Data.csv')
```

# DATASETS

Para python:

```
import pandas as pd  
dataset = pd.read_csv("train.csv")
```

Para Collab:

```
from google.colab import drive  
drive.mount('/content/gdrive')  
dataset = pd.read_csv('gdrive/MyDrive/Data.csv')
```

# DATASETS

En los DataFrames de Pandas existen diferentes formas de seleccionar los registros de las filas y columnas. Siendo dos de las más importantes iloc y loc. La primera permite seleccionar los elementos en base a la posición, mientras que la segunda permite seleccionar mediante etiquetas o declaraciones condicionales.

El método iloc se utiliza en los DataFrames para seleccionar los elementos en base a su ubicación. Su sintaxis es `data.iloc[<filas>, <columnas>]`, donde `<filas>` y `<columnas>` son la posición de las filas y columnas que se desean seleccionar en el orden que aparecen en el objeto.



# DATOS FALTANTES

# DATOS FALTANTES

En los conjuntos de datos del mundo real, los valores faltantes son bastante comunes. Debemos identificar y manejar adecuadamente estos valores faltantes.

En el ámbito del Imputer de Scikit-learn, distinguimos entre imputación numérica e imputación categórica.

La imputación numérica es el proceso de reemplazar los valores numéricos faltantes con estimaciones estadísticas. Es común utilizar la media, la mediana o la moda como valor de reemplazo.

# DATOS FALTANTES

El SimpleImputer de Scikit-learn proporciona una forma sencilla de manejar valores numéricos faltantes. Ofrece varias estrategias, como reemplazar los valores faltantes con la media, la mediana o un valor constante.

```
# Libreria de limpieza
from sklearn.impute import SimpleImputer
# estrategia, media de la columna, es la mas comun (mean), Ojo, NO DE LA FILA
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
```

Al establecer el parámetro de estrategia en 'mean', el SimpleImputer calcula la media de los valores disponibles y reemplaza los valores faltantes con ese valor medio. También se puede utilizar 'median' o 'constant' como estrategia, dependiendo de la naturaleza de sus datos.



# DATOS CATEGÓRICOS

# DATOS CATEGORICOS

Hay que convertir las categorias a numeros! Las categorias no son datos numericos, si lo intentamos meter en una ecuación. No podemos realizar operaciones con textos. Hay que saber traducir a números.

El método de codificación Label Encoding es una forma sencilla de asignar valores numéricos a las diferentes categorías de una variable categórica. Sin embargo, presenta una limitación importante, y es que estos valores numéricos pueden ser malinterpretados por algunos algoritmos de aprendizaje automático. Por ejemplo, si codificamos cuatro ciudades con los valores 0, 1, 2 y 3, es posible que un algoritmo interprete erróneamente que, por ejemplo, la ciudad correspondiente al valor 3 tiene -según algún criterio- un valor tres veces mayor que la ciudad con el valor 1, lo cual no es cierto.

# DATOS CATEGORICOS

Una alternativa al Label Encoding es el método de codificación llamado One Hot Encoding. Esta estrategia consiste en crear una columna binaria (que solo puede contener los valores 0 o 1) para cada valor único que exista en la variable categórica que estamos codificando, y marcar con un 1 la columna correspondiente al valor presente en cada registro, dejando las demás columnas con un valor de 0.

Existe una librería en sklearn que nos ayuda con este tema de conversión. La entrada de este transformador debe ser un array-like de enteros o cadenas, que denote los valores que toman las características categóricas (discretas).

# DATOS CATEGORICOS

Las características se codifican utilizando un esquema de codificación one-hot (también conocido como “one-of-K” o “dummy”). Esto crea una columna binaria para cada categoría y devuelve una matriz dispersa o un arreglo denso (dependiendo del parámetro sparse).

Por defecto, el codificador deriva las categorías basándose en los valores únicos de cada característica. Alternativamente, también puedes especificar las categories manualmente.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
# Columna cero
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
# transformar todo el conjunto X, todas las filas
X = np.array(ct.fit_transform(X))
# Traduce categoria sin orden a un conjunto de tantas columnas como categorías existen
```



## DIVISION DATASET EN TRAIN Y TEST

# DIVISION DATASETS

Los algoritmos de Machine Learning aprenden de los datos con los que los entrenamos. A partir de ellos, intentan encontrar o inferir el patrón que les permita predecir el resultado para un nuevo caso. Pero, para poder calibrar si un modelo funciona, necesitaremos probarlo con un conjunto de datos diferente. Por ello, en todo proceso de aprendizaje automático, los datos de trabajo se dividen en dos partes: datos de entrenamiento y datos de prueba o test.

Los datos de entrenamiento o "training data" son los datos que usamos para entrenar un modelo. La calidad de nuestro modelo de aprendizaje automático va a ser directamente proporcional a la calidad de los datos. Por ello las labores de limpieza, depuración o "data wrangling" consumen un porcentaje importante del tiempo de los científicos de datos.

# DIVISION DATASETS

Los datos de prueba, validación o "testing data" son los datos que nos "reservamos" para comprobar si el modelo que hemos generado a partir de los datos de entrenamiento "funciona". Es decir, si las respuestas predichas por el modelo para un caso totalmente nuevo son acertadas o no.

Es importante que el conjunto de datos de prueba tenga un volumen suficiente como para generar resultados estadísticamente significativos, y a la vez, que sea representativo del conjunto de datos global.

Normalmente el conjunto de datos se suele repartir en un 70% de datos de entrenamiento y un 30% de datos de test, pero se puede variar la proporción según el caso. Lo importante es ser siempre conscientes de que hay que evitar el sobreajuste u "overfitting".

# DIVISION DATASETS

```
from sklearn.model_selection import train_test_split
# Tendremos ahora 4 variables en vez de 2
# random_state numero para reproducir el algoritmo, una semilla aleatoria, que siempre va a dar el mismo resultado
# Cada vez que se ejecuta
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```



# ESCALAMIENTO DE DATOS

# ESCALAMIENTO

En el mundo de la ciencia de datos y el análisis estadístico, la estandarización y la normalización de variables desempeñan un papel fundamental en la preparación y transformación de datos. Estas técnicas nos permiten ajustar y escalar nuestros datos para garantizar una comparabilidad precisa, minimizar el impacto de valores atípicos y mejorar la interpretación de los resultados.

La estandarización y la normalización no son lo mismo, aunque a menudo se utilizan indistintamente y pueden generar cierta confusión. Ambas técnicas buscan ajustar y transformar los datos, pero tienen objetivos y enfoques ligeramente diferentes.

# ESCALAMIENTO

La estandarización se enfoca en la transformación de los datos para que tengan una media de cero y una desviación estándar de uno, mientras que la normalización busca ajustar los datos a un rango específico o a una escala definida. Ambas técnicas son útiles en diferentes situaciones y se seleccionan según los requisitos y objetivos del análisis de datos.

La normalización es una forma de escalar y transformar los datos para que estén en un rango común, independientemente de la escala original de los datos. Esta técnica se utiliza para estandarizar los datos y reducir el impacto de las diferencias en la escala y la magnitud de los atributos de los datos.

# ESCALAMIENTO

Standardisation	Normalisation
$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$	$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$

# ESCALAMIENTO

## Estandarización:

1. Análisis multivariante: En el análisis multivariante, es común estandarizar las variables para garantizar una comparabilidad precisa y equilibrada entre ellas. Esto es especialmente importante en técnicas como el análisis de componentes principales (PCA) y el análisis discriminante lineal (LDA).
2. Algoritmos sensibles a la escala: Algunos algoritmos de aprendizaje automático, como la regresión logística, las máquinas de vectores de soporte (SVM) y el análisis de conglomerados (clustering), son sensibles a la escala de las variables. En estos casos, la estandarización ayuda a equilibrar el impacto de las características y mejora el rendimiento del modelo.
3. Variables con diferentes unidades de medida: Cuando se trabaja con variables que tienen diferentes unidades de medida, como longitud, peso o tiempo, la estandarización es útil para colocarlas en una escala común. Esto facilita la comparación y el análisis de las variables.

# ESCALAMIENTO

## Normalización:

1. Modelos de aprendizaje automático basados en distancias: En algoritmos que utilizan medidas de distancia, como el vecino más cercano (k-NN) y el clustering basado en la distancia, es importante normalizar las variables para que tengan una escala comparable. Esto evita que las variables con rangos más amplios dominen las distancias calculadas.
2. Redes neuronales: En el entrenamiento de redes neuronales, es común normalizar los datos de entrada para que tengan una distribución cercana a una distribución normal, con una media cercana a cero y una desviación estándar de uno. Esto ayuda a que el proceso de entrenamiento sea más estable y eficiente.
3. Análisis de imágenes y procesamiento de señales: En el procesamiento de imágenes y señales, es necesario normalizar los valores de los píxeles o las muestras para que estén dentro de un rango específico, como  $[0, 1]$  o  $[-1, 1]$ . Esto asegura que los datos estén en un formato adecuado para el análisis y la manipulación.

# ESCALAMIENTO

```
from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
X_train[:, 3:] = sc.fit_transform(X_train[:, 3:])  
X_test[:, 3:] = sc.transform(X_test[:, 3:])
```



# EVALUACIÓN

# PREGUNTAS Y RESPUESTAS

Mtro. Alfonso Gregorio Rivero Duarte

Senior Data Manager - CBRE

(+52) 5528997069

devil861109@gmail.com

<https://www.linkedin.com/in/alfonso-gregorio-rivero-duarte-139a9225/>

