**Notita:**

**%matplotlib inline**

Esta instrucción nos permite que nuestros gráficos nos aparezcan en el mismo notebook

**Primero:**

Importamos nuestras librerías

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

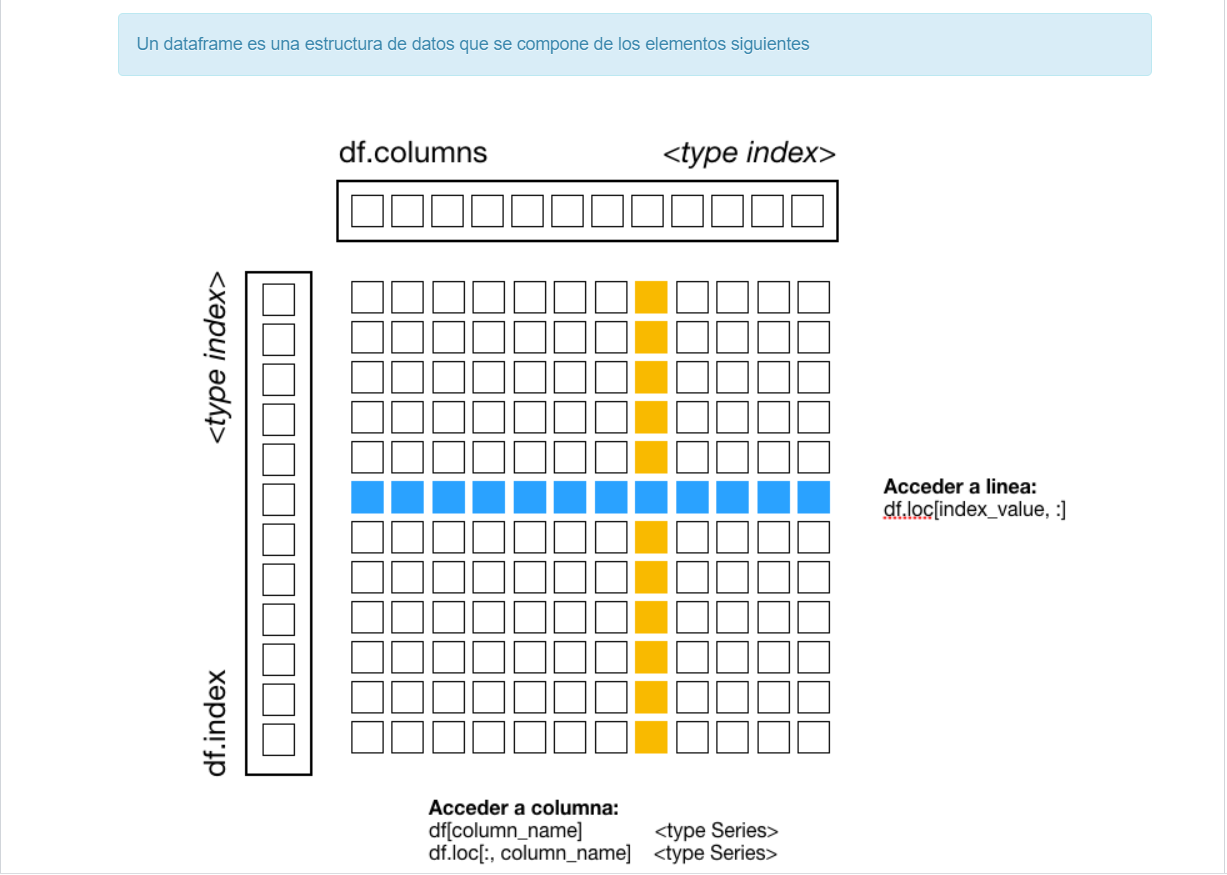
%matplotlib inline

**Segundo:**

Importamos nuestro dataset

movies = **pd.read\_csv(** './Datasets/peliculas.csv' **)**

**Estructura de un dataset**

****

Un DataFrame tiene un index de columnas (la primera fila) y un index de filas (la primera columna) , los cuales nos ayudan a acceder fácilmente a las filas o culumnas o también a los elementos.

**Notita2:**

columna1 = movies**[** 'movie\_title' **]**

#Otra manera de obtener lo mismo podemos, pero de una manera al estilo "numpy"

columna1 = movies**.loc[** : , 'movie\_title' **]**

**Haremos una inspección de nuestros datos**

Lo que haremos ahora es inspecciona nuestros datos. Queremos saber la salud de nuestros datos: Si están limpios o no. Además queremos tener un entendimiento cuantitativo de ellos: Mirar gráficos estadísticos y ver propiedades numéricas de nuestras columnas.

movies.**info( ) :** Este método nos devuelve una vista general de nuestro dataset

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5043 entries, 0 to 5042

Data columns (total 19 columns):

movie\_title 5043 non-null object

title\_year 4935 non-null float64

color 5024 non-null object

aspect\_ratio 4714 non-null float64

duration 5028 non-null float64

language 5031 non-null object

country 5038 non-null object

duration.1 5028 non-null float64

genres 5043 non-null object

content\_rating 4740 non-null object

plot\_keywords 4890 non-null object

director\_name 4939 non-null object

actor\_1\_name 5036 non-null object

actor\_2\_name 5030 non-null object

actor\_3\_name 5020 non-null object

cast\_total\_facebook\_likes 5043 non-null int64

budget 4551 non-null float64

imdb\_score 5043 non-null float64

gross 4159 non-null float64

dtypes: float64(7), int64(1), object(11)

memory usage: 748.6+ KB

Nos indica cuántos datos “ no nulos “ tenemos en cada columna, así como también el tipo de dato de la misma.

**Dato importante**

A diferencia de **Numpy, Pandas** permite cargar no solo datos numéricos pero tambien **datos de texto** que vemos por ejemplo en las columnas de actores y **mezclar distintos tipos de datos**

* int64 y float64 corresponden a los mismos types de Numpy
* object es el dtype que permite manejar datos de texto

Nota: **“object”** es la forma en cómo pandas reconoce los datos de texto.

Ahora, procedemos a separar los datos numéricos de los datos en base a texto

**Numéricos:**

num\_cols = [c for c in movies.columns if (movies[c].dtype == float)| (movies[c].dtype == 'int64')]

movies\_num = movies[num\_cols]

movies\_num.head()

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **title\_year** | **aspect\_ratio** | **duration** | **duration.1** | **budget** | **imdb\_score** | **gross** |
| **0** | 2009.0 | 1.78 | 178.0 | 178.0 | 237000000.0 | 7.9 | 760505847.0 |

**Inspección cuantitativa y de salud de los datos**

**Estadísticas**

La instrucción **“ .describe( )** “ nos retorna las propiedades estadísticas de la columna si esta es una numérica. Si no es numérica, nos dará menos información.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **title\_year** | **aspect\_ratio** | **duration** | **duration.1** | **cast\_total\_facebook\_likes** | **budget** | **imdb\_score** | **gross** |
| **count** | **4935.000000** | **4714.000000** | **5028.000000** | **5028.000000** | **5043.000000** | **4.551000e+03** | **5043.000000** | **4.159000e+03** |
| **mean** | **2002.470517** | **2.220403** | **107.201074** | **107.201074** | **9699.063851** | **3.975262e+07** | **6.442138** | **4.846841e+07** |
| **std** | **12.474599** | **1.385113** | **25.197441** | **25.197441** | **18163.799124** | **2.061149e+08** | **1.125116** | **6.845299e+07** |
| **min** | **1916.000000** | **1.180000** | **7.000000** | **7.000000** | **0.000000** | **2.180000e+02** | **1.600000** | **1.620000e+02** |
| **25%** | **1999.000000** | **1.850000** | **93.000000** | **93.000000** | **1411.000000** | **6.000000e+06** | **5.800000** | **5.340988e+06** |
| **50%** | **2005.000000** | **2.350000** | **103.000000** | **103.000000** | **3090.000000** | **2.000000e+07** | **6.600000** | **2.551750e+07** |
| **75%** | **2011.000000** | **2.350000** | **118.000000** | **118.000000** | **13756.500000** | **4.500000e+07** | **7.200000** | **6.230944e+07** |
| **max** | **2016.000000** | **16.000000** | **511.000000** | **511.000000** | **656730.000000** | **1.221550e+10** |  |  |

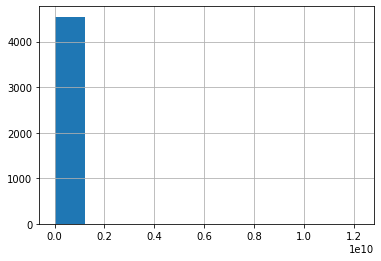
**Notita:** Pandas tiene un método que nos permite ver un histograma sin tener que recurrir a matplotlib

movies\_num**[** 'duration' **].hist()**

**Tuvimos un problema:**

Tuvimos una columna con datos que no estaban estandarizados. Eso suele suceder y es nuestro trabajo limpiar y procesar los datos a mano.

En este caso era una columna con el costo de producción de la película, la cual en sus datos tenía monedas diferentes, lo que provocaba algunos errores al querer visualizar los datos.



Como solución, debemos rehacer todo ese dataset y cambiar a una moneda, por ejemplo en dólares.

**Dato:**

Por otro lado, si queremos hacer un join de una columna a un dataset podemos hacer lo siguiente.

***movies\_num*** era nuestro dataset y ***movies['movie\_title']*** era la columna de nombres que queremos añadir

pd.concat([movies\_num, movies['movie\_title']], axis = 1) # por defecto se concatena line por linea " axis = 0 ", pero si lo queremos es concatenar

# columna por columna escribimos " axis = 1 "

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **title\_year** | **aspect\_ratio** | **duration** | **duration.1** | **cast\_total\_facebook\_likes** | **budget** | **imdb\_score** | **gross** | **movie\_title** |
| **0** | 2009.0 | 1.78 | 178.0 | 178.0 | 4834 | 237000000.0 | 7.9 | 760505847.0 | Avatar |

Ahora lo que haremos es una fusión de los datasets ***financial*** y ***movies\_num*** con “**pd.merge**”

#Hacemos la fusión de los dataframes "financials" y "movies\_num" con "pd.merge()"

#le pasamos los 2 dataframes a juntar , a "on" le pasamos la clave que nos va a relacionar ambos dataframes, porque aparece en ambas.

# a "how" le decimos que el largo debe de ser como el largo de las columnas de las claves de el dataset izquierdo

movies\_v2 = pd.merge(financials, movies\_num, on= 'movie\_title', how ='left')

movies\_v2.head()

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **movie\_title** | **production\_budget** | **worldwide\_gross** | **title\_year** | **aspect\_ratio** | **duration** | **duration.1** | **cast\_total\_facebook\_likes** | **budget** | **imdb\_score** | **gross** |  |
| **0** | Avatar | 425000000 | 2783918982 | 2009.0 | 1.78 | 178.0 | 178.0 | 4834 | 237000000.0 | 7.9 | 760505847.0 |

**Manejo de datos faltantes**

movies\_v2**.notnull( )** : Nos retorna un booleano indicando si hay un número faltante o no.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **movie\_title** | **production\_budget** | **worldwide\_gross** | **title\_year** | **aspect\_ratio** | **duration** | **duration.1** | **cast\_total\_facebook\_likes** | **budget** | **imdb\_score** | **gross** |
| **0** | True | True | True | True | True | True | True | True | True | True |

# "pd.Series.value\_counts" lo que hace es contar en una columna cuántos elementos hay del mismo valor. En este caso, tenemos True o False

#"apply" es una función que nos evita hacer un for loop manualmente

#Ahora lo que va a hacer el "apply" es que columna por columna va a aplicar "pd.Serie.value\_counts" y con ello obtendremos cuántos elementos nulos tenemos en cada columna

movies\_v2.notnull().apply(pd.Series.value\_counts)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| **movie\_title** | **production\_budget** | **worldwide\_gross** | **title\_year** | **aspect\_ratio** | **duration** | **duration.1** | **cast\_total\_facebook\_likes** | **budget** |  |  |  |
| **False** | NaN | NaN | NaN | 27 | 274 | 13 | 13 | NaN |  |  |  |
| **True** | 4385.0 | 4385.0 | 4385.0 | 4358 | 4111 | 4372 | 4372 | 4385.0 |  |  |  |

#Ahora haremos lo mismo que antes para poder ver las filas que al menos tengan un valor nulo o vacío

#Primero a available le pasamos todo el dataframe con indicaciones de true si es un valor no nulo o no vacío

available = ((movies\_v2 !=0) & (movies\_v2.notnull()))

available.head()

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **movie\_title** | **production\_budget** | **worldwide\_gross** | **title\_year** | **aspect\_ratio** | **duration** | **duration.1** | **cast\_total\_facebook\_likes** | **budget** | **imdb\_score** | **gross** |
| **0** | True | True | True | True | True | True | True | True | True | True |
| **1** | True | True | True | False | False | False | False | True | False | True |