

# ANÁLISIS Y EXPLORACIÓN DE DATOS

Versión : 1.2, Para uso educativo  
Autor : Luis Beltran Palma Ttito  
Lugar : Cusco, Perú, 2023.  
Proposito : Exploración de datos de 'Insuficiencia cardiaca', con estadística descriptiva

## ▼ 1. PASOS PREVIOS

### ▼ Importar librería

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt # https://matplotlib.org/stable/index.html
4 import seaborn as sns # https://seaborn.pydata.org/
5
6 # muestra el gráfico en el cuaderno (no utiliza una ventana)
7 %matplotlib inline
```

### ▼ Descargar dataset

```
1 !wget https://archive.ics.uci.edu/ml/machine-learning-databases/00519/heart_failure_clinical_records_dataset.csv
--2023-05-09 17:27:59-- https://archive.ics.uci.edu/ml/machine-learning-databases/00519/heart\_failure\_clinical\_records\_dataset.csv
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12239 (12K) [application/x-httdp-php]
Saving to: 'heart_failure_clinical_records_dataset.csv.1'

heart_failure_clini 100%[=====] 11.95K --.-KB/s   in 0s
2023-05-09 17:27:59 (81.0 MB/s) - 'heart_failure_clinical_records_dataset.csv.1' saved [12239/12239]
```

### ▼ Cargar dataset con pandas

```
1 datos = pd.read_csv('heart_failure_clinical_records_dataset.csv', delimiter=',', header=0)
2 datos.head()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_s
0	75.0	0	582	0	20		1	265000.00	1.9
1	55.0	0	7861	0	38		0	263358.03	1.1
2	65.0	0	146	0	20		0	162000.00	1.3
3	50.0	1	111	0	20		0	210000.00	1.9
4	65.0	1	160	1	20		0	327000.00	2.7

#### Descripción del dataset:

**age (edad):** Edad del paciente (años)

**anaemia(anemia):** 0=NO, 1=SÍ; Disminución de glóbulos rojos o hemoglobina (valor booleano).

**high\_blood\_pressure (hipertensión arterial):** 0=NO, 1=SÍ; Si el paciente tiene presión arterial alta o no.

**creatinine\_phosphokinase (creatinina fosfoquinasa) - CPK:** nivel de la enzima CPK en la sangre (mcg/L)

**diabetes:** 0=NO, 1=SÍ; El nivel de glucosa en sangre en ayunas en individuos sanos está en el rango de 70-100 mg/dL. Si el nivel de azúcar en la sangre excede este rango, generalmente indica diabetes.

**ejection\_fraction (fracción de eyección):** el porcentaje de sangre que se bombea del corazón; lo normal es 55% a más.

**platelets (plaquetas):** es el nombre que se le da a las células involucradas en la formación de coágulos sanguíneos. El valor normal de trombocito en sangre es de 150 mil entre 450 mil en un mm<sup>3</sup> de sangre.

**sex (sexo):** 0=MUJER , 1=MASCULINO; Género del paciente.

**serum\_creatinine (creatinina sérica):** Nivel de creatina producida por los riñones en la sangre.

**serum\_sodium (sodio sérico):** Nivel de creatinina sérica en la sangre. El rango normal de sodio en la sangre es de 135-145 mEq/L. Una situación con más de 145 milimoles de sodio en un litro de sangre significa que el cuerpo pierde más agua de la que ingresa al cuerpo.

**smoking (tabaquismo):** 0=NO FUMA, 1=SI FUMA.

**time (tiempo):** Período de seguimiento (días)

**DEATH\_EVENT (Evento de muerte):** 0=NO, 1=SI; si el paciente falleció durante el período de seguimiento (booleano). Es el atributo objetivo.

## ▼ 2. EXPLORANDO EL DATASET

### ▼ Información sobre la estructura del dataset

```
1 datos.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   age              299 non-null    float64
 1   anaemia          299 non-null    int64  
 2   creatinine_phosphokinase 299 non-null    int64  
 3   diabetes         299 non-null    int64  
 4   ejection_fraction 299 non-null    int64  
 5   high_blood_pressure 299 non-null    int64  
 6   platelets        299 non-null    float64
 7   serum_creatinine 299 non-null    float64
 8   serum_sodium     299 non-null    int64  
 9   sex              299 non-null    int64  
 10  smoking          299 non-null    int64  
 11  time             299 non-null    int64  
 12  DEATH_EVENT      299 non-null    int64  
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

**IMPORTANTE:** El dataset posee 13 atributos numéricos y cada atributo posee 299 datos. **No hay datos faltantes**

### ▼ Detección de valores nulos

```
1 datos.isnull().sum()

age                  0
anaemia              0
creatinine_phosphokinase 0
diabetes              0
ejection_fraction    0
high_blood_pressure  0
platelets            0
serum_creatinine     0
serum_sodium          0
sex                  0
smoking              0
time                 0
DEATH_EVENT          0
dtype: int64
```

### ▼ Estadística descriptiva de los datos

```
1 # descripción estadística del dataset
2 datos.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
<b>age</b>	299.0	60.833893	11.894809	40.0	51.0	60.0	70.0	95.0
<b>anaemia</b>	299.0	0.431438	0.496107	0.0	0.0	0.0	1.0	1.0
<b>creatinine_phosphokinase</b>	299.0	581.839465	970.287881	23.0	116.5	250.0	582.0	7861.0
<b>diabetes</b>	299.0	0.418060	0.494067	0.0	0.0	0.0	1.0	1.0
<b>ejection_fraction</b>	299.0	38.083612	11.834841	14.0	30.0	38.0	45.0	80.0
<b>high_blood_pressure</b>	299.0	0.351171	0.478136	0.0	0.0	0.0	1.0	1.0
<b>platelets</b>	299.0	263358.029264	97804.236869	25100.0	212500.0	262000.0	303500.0	850000.0
<b>serum_creatinine</b>	299.0	1.393880	1.034510	0.5	0.9	1.1	1.4	9.4
<b>serum_sodium</b>	299.0	136.625418	4.412477	113.0	134.0	137.0	140.0	148.0
<b>sex</b>	299.0	0.648829	0.478136	0.0	0.0	1.0	1.0	1.0
<b>smoking</b>	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0
<b>time</b>	299.0	130.260870	77.614208	4.0	73.0	115.0	203.0	285.0
<b>DEATH_EVENT</b>	299.0	0.321070	0.467670	0.0	0.0	0.0	1.0	1.0

## ▼ Filtros

```
1 # nombre de las columnas del dataset
2 datos.columns
```

```
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
       'DEATH_EVENT'],
      dtype='object')
```

```
1 datos.head()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sod
0	75.0	0	582	0	20		1	265000.00	1.9
1	55.0	0	7861	0	38		0	263358.03	1.1
2	65.0	0	146	0	20		0	162000.00	1.3
3	50.0	1	111	0	20		0	210000.00	1.9
4	65.0	1	160	1	20		0	327000.00	2.7

```
1 # Filtros con pandas: lista de varones
2 filtro_varones = datos['sex'] == 1
3 varones = datos[filtro_varones]
4 varones.head()
5
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sod
0	75.0	0	582	0	20		1	265000.00	1.9
1	55.0	0	7861	0	38		0	263358.03	1.1
2	65.0	0	146	0	20		0	162000.00	1.3
3	50.0	1	111	0	20		0	210000.00	1.9
5	90.0	1	47	0	40		1	204000.00	2.1

```
1 # lista de mujeres
2 mujeres = datos[datos.sex == 0]
3 mujeres.head()
4
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_s
4	65.0	1		160	1	20	0	327000.00	2.7
8	65.0	0		157	0	65	0	263358.03	1.5
14	49.0	1		80	0	30	1	427000.00	1.0

```
1 # varones monitoreados por más de 250 días
2 varones250 = datos[(datos.time > 250) & (datos.sex == 1)]
3 varones250.head()
4
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_s
291	60.0	0		320	0	35	0	133000.0	1.4
292	52.0	0		190	1	38	0	382000.0	1.0
293	63.0	1		103	1	35	0	179000.0	0.9
294	62.0	0		61	1	38	1	155000.0	1.1
297	45.0	0		2413	0	38	0	140000.0	1.4

```
1 #pacientes no fumadores monitoreados por más de 250 días
2 NoFumadores = datos[(datos['time'] > 250) & ( datos['smoking'] == 0) ]
3 NoFumadores.head()
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_s
288	65.0	0		892	1	35	0	263358.03	1.1
289	90.0	1		337	0	38	0	390000.00	0.9
290	45.0	0		615	1	55	0	222000.00	0.8
291	60.0	0		320	0	35	0	133000.00	1.4
295	55.0	0		1820	0	38	0	270000.00	1.2



## ▼ 3. VISTA GRÁFICA

¿Qué se busca en los gráficos?

- Valores atípicos (Outliers)
- Asimetría de la distribución
- Cambios en variabilidad
- No-linealidad

## ▼ Histograma

Un histograma es un ejemplo de un gráfico de densidad, es decir cada barra describe la frecuencia, porcentaje o densidad de los valores que se incluyen en los datos entre el límite inferior y el límite superior de cada barra

VENTAJA:

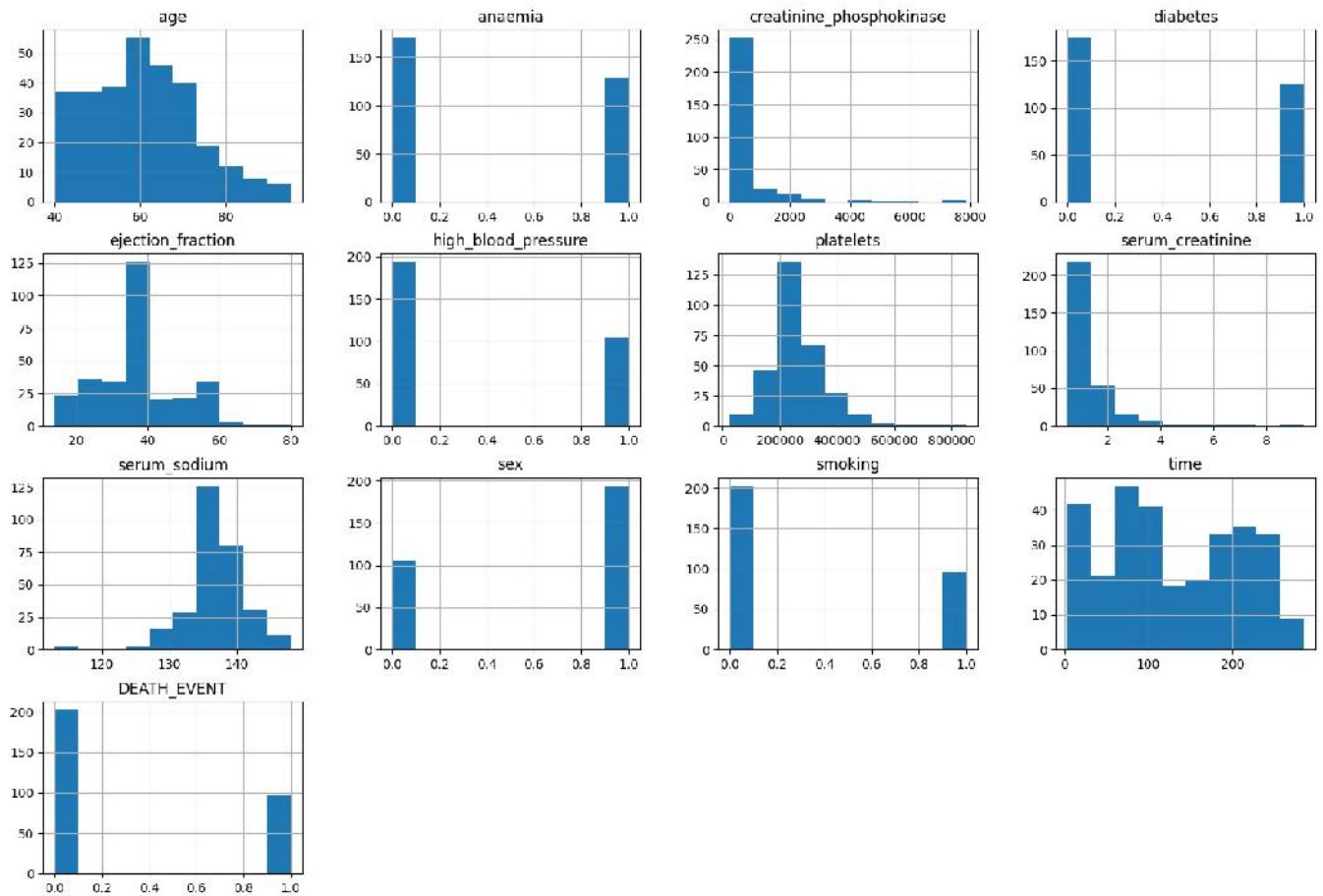
Es útil para apreciar la forma de la distribución de los datos, si se escoge adecuadamente el número de clases y su amplitud.

DESVENTAJAS:

- Las observaciones individuales se pierden.
- La división en clases o intervalos es arbitraria.
- El cambiar el número de intervalos cambia la forma de la distribución.
- No pueden derivarse estadísticos básicos.

```
1 # histograma por cada atributo del dataset
2 datos.hist(figsize=(18,12))
```

```
3 plt.show()
```



#### Algunas observaciones interesantes:

Edad: Gran porcentaje de la población está entre 40 y 70 años

Plaquetas: La mayoría de población posee la cantidad normal de plaquetas (entre 150000 y 400000)

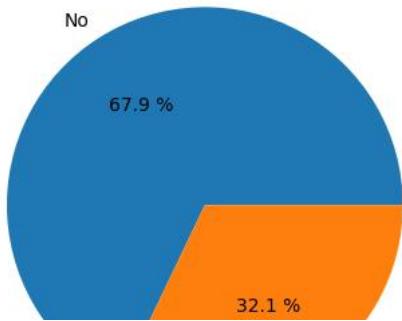
¿Qué otros puntos saltan a la vista?

#### ▼ Pastel: Distribución de clases

```
1 # agrupación de atributo objetivo  
2 datos.groupby('DEATH_EVENT').size()
```

```
DEATH_EVENT  
0    203  
1     96  
dtype: int64
```

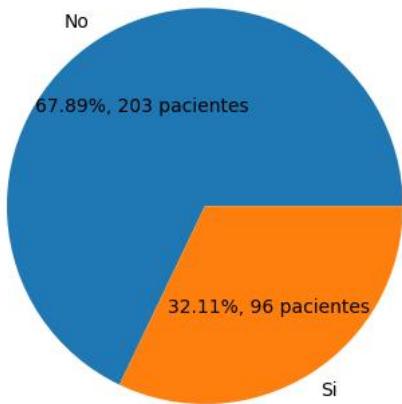
```
1 datos.columns  
  
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',  
       'ejection_fraction', 'high_blood_pressure', 'platelets',  
       'serum_CREATININE', 'serum_sodium', 'sex', 'smoking', 'time',  
       'DEATH_EVENT'],  
      dtype='object')  
  
1 # generar gráfico de pastel  
2 plt.pie(datos.groupby('DEATH_EVENT').size(), autopct="%0.1f %%", labels=['No','Si'])  
3 plt.show()
```



```

1 # generar gráfico de pastel incluyendo valores y porcentajes
2 values = datos.groupby('DEATH_EVENT').size()
3 etiquetas = ['No', 'Si']
4 plt.pie(values, autopct=lambda p:f'{p:.2f}%, {p*sum(values)/100 :.0f} pacientes',
5           labels=etiquetas)
6 plt.show()

```



## ▼ Diagramas de dispersión

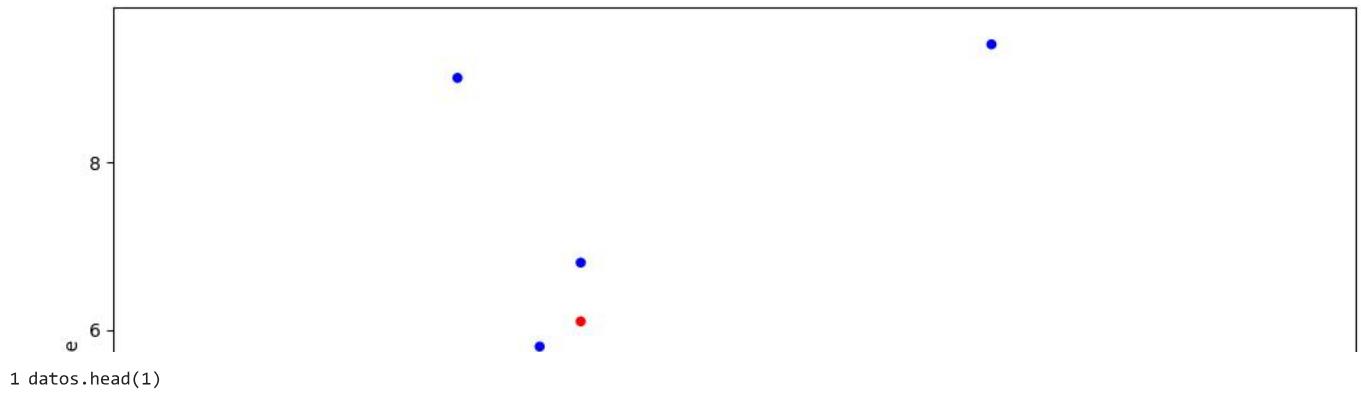
Permiten identificar relaciones no-lineales y outliers.

```

1 # configurar colores para las clases de DEATH_EVENT
2 set(datos.DEATH_EVENT)
3 color={1:'blue',0:'red'}
4 datos_color = datos.DEATH_EVENT.map(color)

1 # graficar los puntos: creatinina sérica vs edad
2 datos.plot("age", "serum_creatinine", kind="scatter", color=datos_color, figsize=(12,8))
3 plt.show()

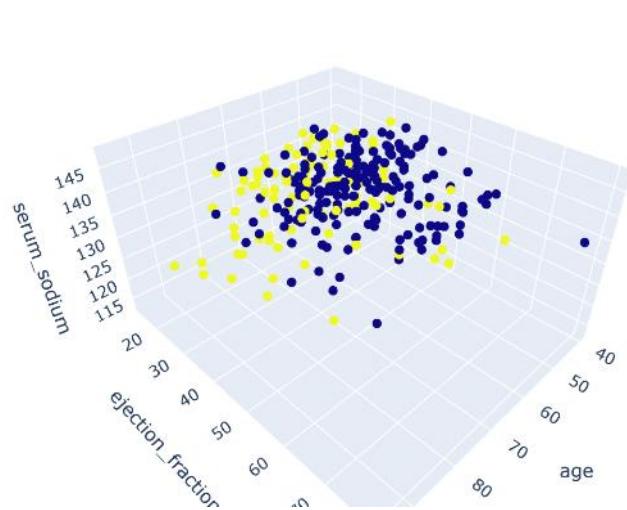
```



```
1 datos.head(1)
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium
0	75.0	0	582	0	20	1	265000.0	1.9	

```
1 # diagrama de dispersión en 3D: edad vs. fracción de eyección vs. sodio sérico
2 import plotly.express as px
3 fig = px.scatter_3d(datos, x='age', y='ejection_fraction',
4                      z='serum_sodium', color='DEATH_EVENT')
5 fig.update_traces(marker_size = 4)
6 fig.show()
```

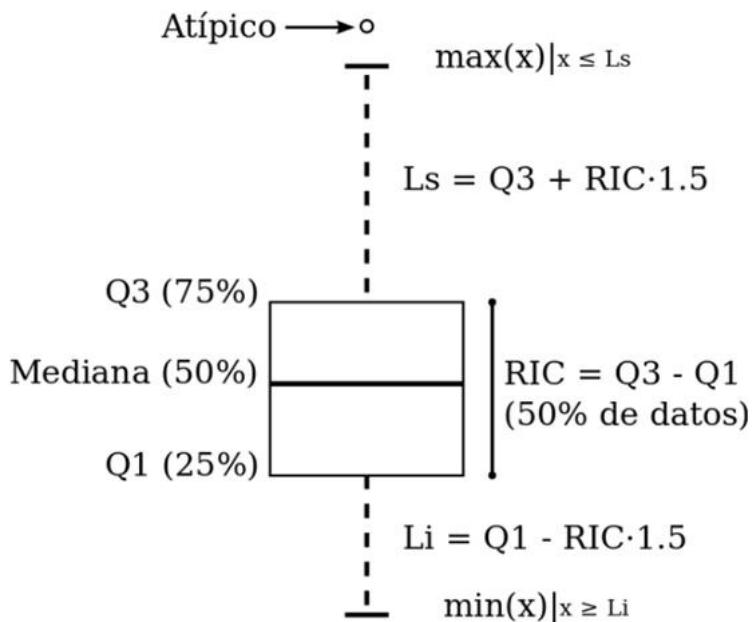


## ▼ Diagrama de caja (boxplot)

Efectivo para tamaños de muestra  $\geq 8$ . Provee más estadísticos básicos que el histograma.

VENTAJAS:

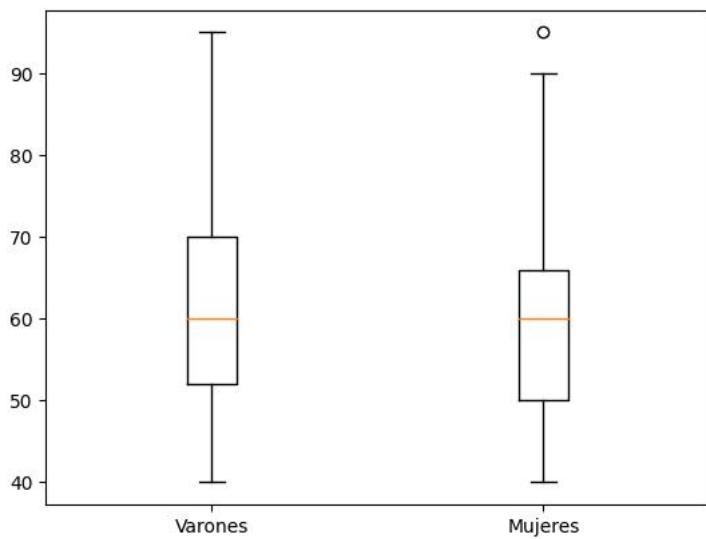
- || Como se basan en la mediana, son robustos a valores atípicos.
- || Indican la variabilidad de los datos por la distancia entre los bigotes.
- || Indican la forma de la distribución, especialmente si es simétrica o asimétrica.
- || Permiten la comparación de varios grupos simultáneamente.
- || Indican la presencia de outliers (valores extremos).



```

1 # diagrama de caja por sexo y edad
2 plt.boxplot([varones.age, mujeres.age], labels=['Varones','Mujeres'])
3 plt.show()

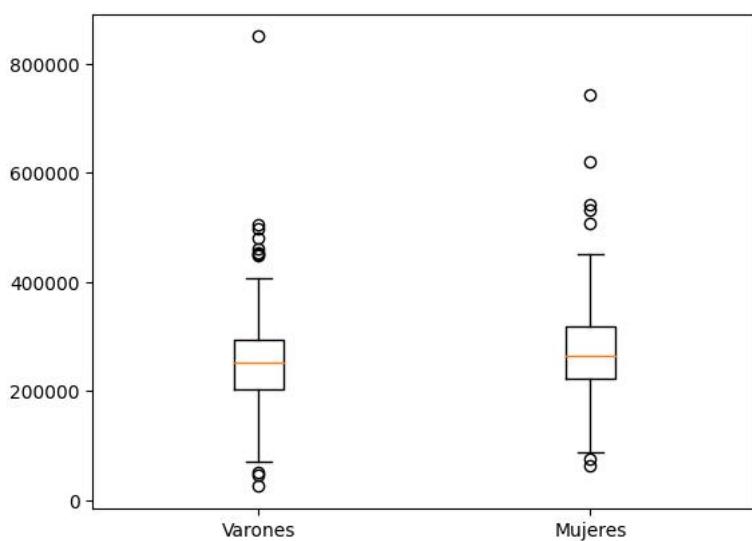
```



```

1 # diagrama de caja por sexo y plaquetas
2 plt.boxplot([varones.platelets, mujeres.platelets], labels=['Varones','Mujeres'])
3 plt.show()

```

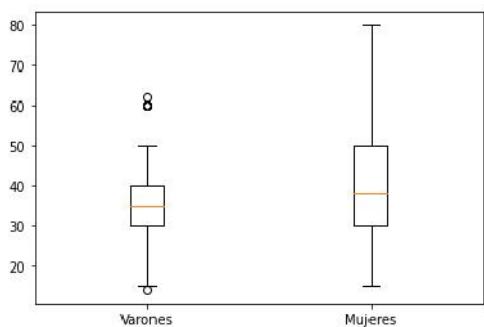


```

1 # diagrama de caja por sexo y fraccion de eyección
2 plt.boxplot([varones.ejection_fraction, mujeres.ejection_fraction], labels=['Varones','Mujeres'])
3 plt.show()

/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDeprecationWarning:
Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths

```



## ▼ 4. ANÁLISIS DE CORRELACIÓN

### ▼ Correlación lineal

**Gráficos Cuantil-Cuantil (Q-Q plots)** Un gráfico Cuantil-Cuantil permite observar cuan cerca está la distribución de un conjunto de datos a alguna distribución ideal ó comparar la distribución de dos conjuntos de datos.

#### Utilidad de los gráficos Q-Q

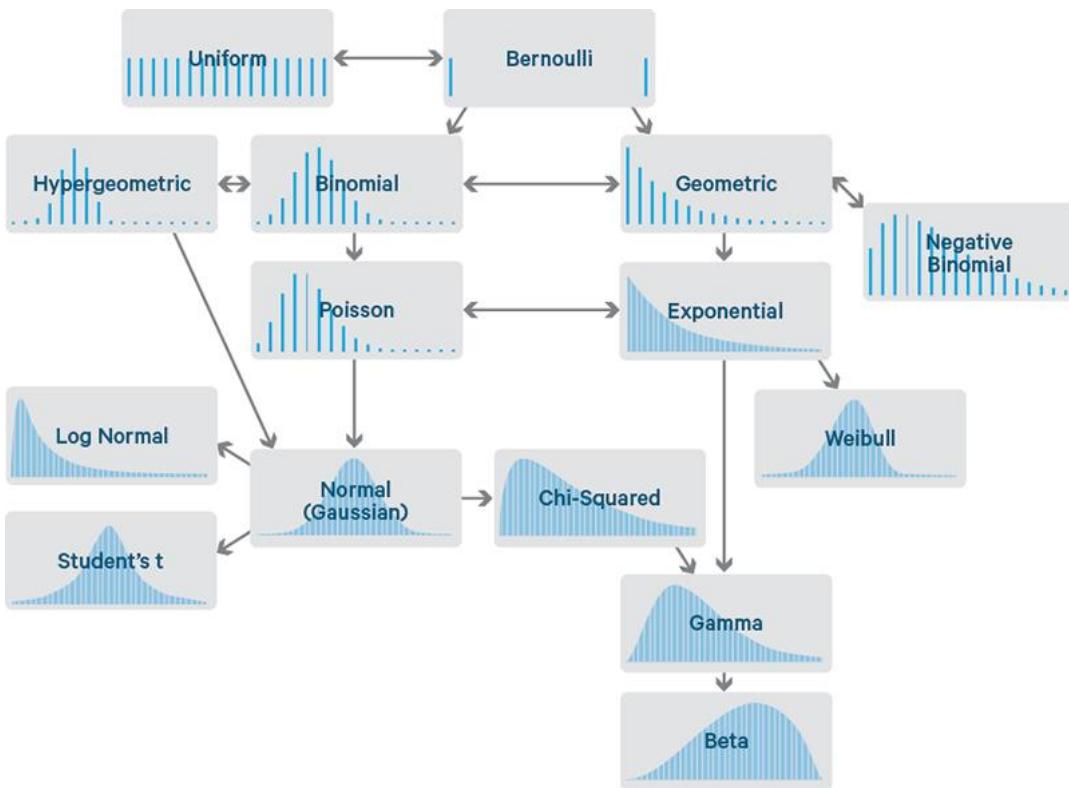
Un gráfico Q-Q es una herramienta útil para evaluar:

1. Si los datos siguen una distribución teórica o si hay desviaciones significativas.
2. Identifica valores atípicos.
3. Permite comparar varias distribuciones entre si.
4. En caso que los datos, sigan una distribución teórica, es posible generar datos artificialmente.

#### Graficando un Q-Q PLOT

Para crear un gráfico Q-Q, se procede de la siguiente manera:

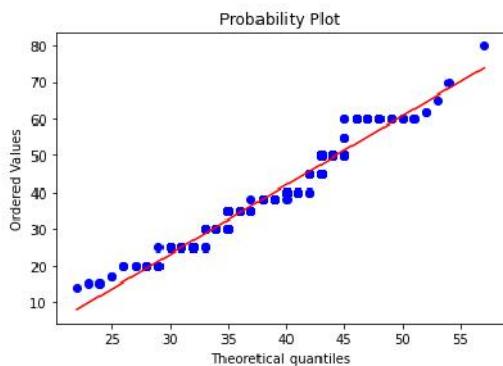
1. Se debe contar, con dos conjuntos de datos X, Y.
2. Se obtienen los cuantiles de X e Y
3. Se crea pares de cuantiles de X e Y
4. Se gráfica los pares de cuantiles
5. Si los conjuntos de datos X e Y son similares, los valores de cuantiles sería tambien similares, ello implica que los pares de cuantiles graficados sigan la linea de la función de identidad.



```

1 # ejemplo:distribución de Poisson
2 import statistics
3 import pylab
4 import scipy.stats as stats
5
6 stats.probplot(datos['ejection_fraction'],
7                  dist=stats.poisson(statistics.median(datos['ejection_fraction'])), plot=pylab)
8 pylab.show()

```

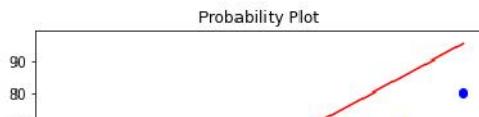


La gráfica Q-Q, compara los valores del atributo `ejection_fraction` con la distribución de poisson. Ya que, hay aproximación de los puntos (azules) a la linea de la función de identidad, podemos afirmar que el atributo `ejection_fraction`, sigue una distribución de poisson.

```

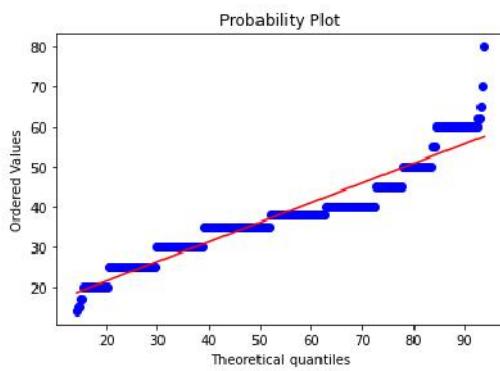
1 # ejemplo:distribución exponencial
2 import statistics
3 import pylab
4 import scipy.stats as stats
5
6 stats.probplot(datos['ejection_fraction'],
7                  dist=stats.expon(statistics.median(datos['ejection_fraction'])), plot=pylab)
8 pylab.show()

```



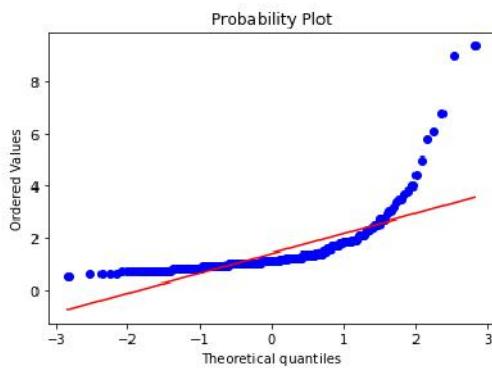
La gráfica Q-Q, compara los valores del atributo *ejection\_fraction* con la distribución expónencial. Ya que no se observa mucha aproximación de los puntos (azules) a la línea de la función de identidad, podemos afirmar que el atributos *ejection\_fraction*, NO sigue una distribución exponencial.

```
30 | 
1 # ejemplo:distribución uniforme
2 import statistics
3 import pylab
4 import scipy.stats as stats
5
6 stats.probplot(datos['ejection_fraction'],
7                  dist=stats.uniform(min(datos['ejection_fraction']),
8                                      max(datos['ejection_fraction'])), plot=pylab)
9 pylab.show()
```



¿Qué opinión le merece el resultado?

```
1 # ejemplo:distribución normal
2 import pylab
3 import scipy.stats as stats
4
5 stats.probplot(datos['serum_creatinine'],dist=stats.norm, plot=pylab)
6 pylab.show()
```

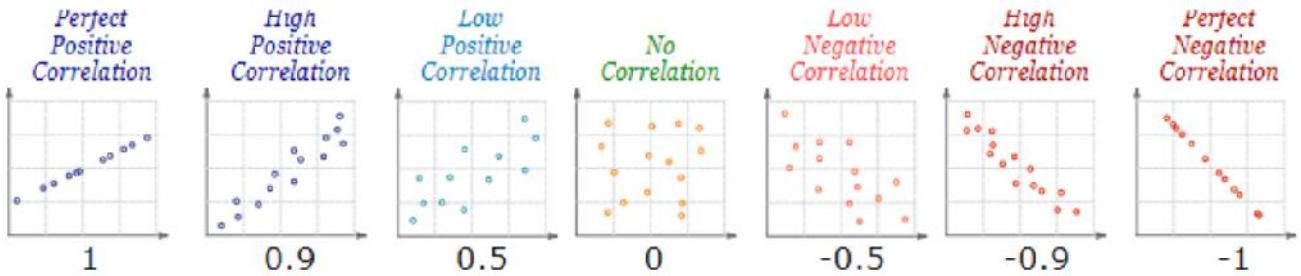


La gráfica Q-Q, compara los valores del atributo *serum\_creatinine* con la distribución normal, y podemos afirmar que los valores del atributo *serum\_creatinine*, NO sigue una distribución normal.

## ▼ Correlación Pearson

La correlación es una medida de la relación (covariación) lineal entre dos variables cuantitativas continuas ( $x, y$ ). La manera más sencilla de saber si dos variables están correlacionadas es determinar si co-varian (varían conjuntamente)

La correlación es un tipo de asociación entre dos variables numéricas, específicamente evalúa la tendencia (creciente o decreciente) en los datos.



El valor del índice de correlación varía en el intervalo [-1,1]

Si este coeficiente es igual a 1 o -1 (o cercano a estos valores) significa que una variable es fruto de una transformación lineal de la otra. Teniendo una relación directa al tratarse de 1 (cuando una variable aumenta, la otra también), mientras que existirá una relación inversa al tratarse de -1 (cuando una variable aumenta la otra disminuye).

Mientras que, Si  $r = 0$  (o cercano a este valor) no existe relación lineal, aunque puede existir algún otro tipo de relación no lineal.

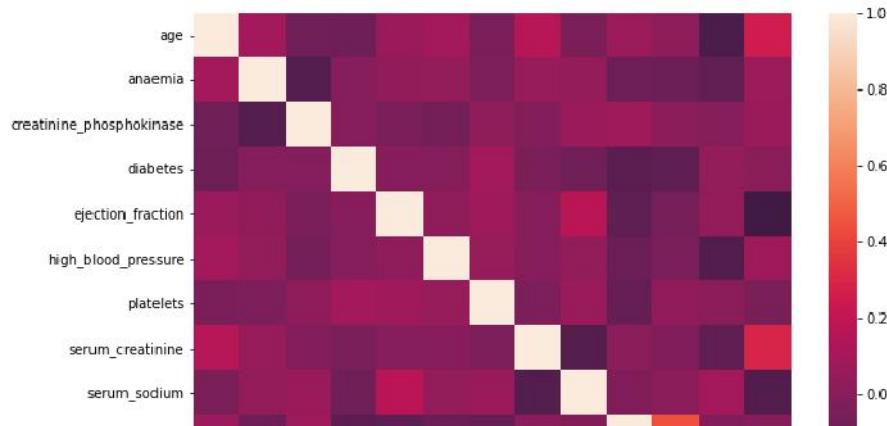
$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

```
1 # Matriz de correlación
2 datos_pearson = datos.corr(method='pearson')
3 datos_pearson
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	
age	1.000000	0.088006		-0.081584	-0.101012	0.060098	0.093289	-0.052354
anaemia	0.088006	1.000000		-0.190741	-0.012729	0.031557	0.038182	-0.043786
creatinine_phosphokinase	-0.081584	-0.190741		1.000000	-0.009639	-0.044080	-0.070590	0.024463
diabetes	-0.101012	-0.012729		-0.009639	1.000000	-0.004850	-0.012732	0.092193
ejection_fraction	0.060098	0.031557		-0.044080	-0.004850	1.000000	0.024445	0.072177
high_blood_pressure	0.093289	0.038182		-0.070590	-0.012732	0.024445	1.000000	0.049963
platelets	-0.052354	-0.043786		0.024463	0.092193	0.072177	0.049963	1.000000
serum_creatinine	0.159187	0.052174		-0.016408	-0.046975	-0.011302	-0.004935	-0.041198
serum_sodium	-0.045966	0.041882		0.059550	-0.089551	0.175902	0.037109	0.062125
sex	0.065430	-0.094769		0.079791	-0.157730	-0.148386	-0.104615	-0.125120
smoking	0.018668	-0.107290		0.002421	-0.147173	-0.067315	-0.055711	0.028234
time	-0.224068	-0.141414		-0.009346	0.033726	0.041729	-0.196439	0.010514
DEATH_EVENT	0.253729	0.066270		0.062728	-0.001943	-0.268603	0.079351	-0.049139

```
1 # gráfico de calor (heatmap) para la correlación de atributos
2 import seaborn as sns
3 fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(10,8))
4 sns.heatmap(datos_pearson,ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2ea46531d0>
```



```
1 # un mapa de calor más refinado
2 corr_pearson = datos.corr()
3 mask = np.array(corr_pearson)
4 mask[np.tril_indices_from(mask)] = False
5 fig,ax= plt.subplots()
6 fig.set_size_inches(20,10)
7 sns.heatmap(corr_pearson, mask=mask,vmax=.8, square=True,annot=True)
```

```
<Axes: >
```

