

```
In [1]: import pandas as pd
import numpy as np
import scipy as scipy
import matplotlib.pyplot as plt
import seaborn as sb
import scipy.stats as stats
from sklearn import linear_model
```

Tipologia i cicle de vida de les dades

Pràctica 2: Com realitzar la neteja i l'anàlisi de dades?

1. Descripció del dataset

Per la realització d'aquesta pràctica s'ha optat per treballar amb el dataset proposat, Heart Attack Analysis & Prediction. Es pot trobar en la direcció següent: <https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset>.

S'anomena atac cardíac al bloqueig del flux de sang al cor. Això pot tenir causes diverses, sent les més comuna la formació de coàguls a les artèries degut a una acumulació de substàncies a la paret de les mateixes. Com més temps passa sense flux de sang, més danys rep l'òrgan.

Amb aquest dataset podem respondre infinitat de preguntes, ja que és molt complet i disposa de moltes columnes que es poden analitzar. Algunes de les preguntes que podem respondre són les següents:

- L'edat influeix en el risc d'atac cardíac?
- Qui té més risc, homes o dones?
- Quins valors/síntomes són més rellevants/indicatius en un atac cardiovascular?
- Correlació entre variables

El dataset conté 14 variables, que presenten la descripció següent (extreta de Kaggle):  Description

Aquesta descripció ens serviria properament a manera de diccionari per tal de comprovar possibles falles en el dataset, com podrien ser classes de més en alguna variable o valors fora d'un rang preestablert.

2. Integració i selecció

Per aquesta pràctica s'ha escollit no eliminar cap de les columnes ja presents ni agregar-ne cap de nova. Considero que el dataset és molt complet i totes les variables que presenta poden ser utilitzades per al seu estudi. De totes maneres, l'abast d'aquesta pràctica no és la implementació de cap model complex que utilitzi totes les característiques. Tot i així, en un cas real on volguéssim fer un anàlisi profund no eliminaríem cap columna.

3. Neteja de les dades

3.1 Les dades contenen zeros o elements buits?

Comencem llegint el dataset i mostrant-ne les primeres files, així com la seva forma i informació de cada una de les columnes:

```
In [2]: df = pd.read_csv("heart.csv")
df.head()

Out[2]:
```

	age	sex	cp	trtbps	chol	fb	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
In [3]: df.shape

Out[3]: (383, 14)
```

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 383 entries, 0 to 382
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  --
 0   age        383 non-null    int64
 1   sex        383 non-null    int64
 2   cp         383 non-null    int64
 3   trtbps     383 non-null    int64
 4   chol       383 non-null    int64
 5   fbs        383 non-null    int64
 6   restecg    383 non-null    int64
 7   thalachh   383 non-null    int64
 8   exng       383 non-null    int64
 9   oldpeak    383 non-null    float64
10   slp        383 non-null    int64
11   caa        383 non-null    int64
12   thall      383 non-null    int64
13   output     383 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Com observem amb ajuda del mètode .info(), en cap columna tenim cap valor buit, ja que tenim 383 registres en total i tots són non-null. El que sí que veiem, però, és que hi ha certes columnes amb data types incorrectes, com ara "cp", que representa el tipus de dolor de pit. Si bé és veritat que es representa com a un número (de 0 a 3), és una columna categòrica, ja que no pot presentar un valor fora d'aquest rang. El mateix passa amb les columnes "sex", "fbs", "restecg", "exng", "slp", "thall", "caa" i "output".

Corregim aquests data types:

```
In [5]: cat_columns = ["sex", "cp", "fbs", "restecg", "exng", "slp", "thall", "caa", "output"]
flt_columns = ["age", "trtbps", "chol", "thalachh", "oldpeak"]
df[cat_columns] = df[cat_columns].astype("category")
df[flt_columns] = df[flt_columns].astype("float64")
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 383 entries, 0 to 382
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  --
 0   age        383 non-null    float64
 1   sex        383 non-null    category
 2   cp         383 non-null    category
 3   trtbps     383 non-null    float64
 4   chol       383 non-null    float64
 5   fbs        383 non-null    category
 6   restecg    383 non-null    category
 7   thalachh   383 non-null    float64
 8   exng       383 non-null    category
 9   oldpeak    383 non-null    float64
10   slp        383 non-null    category
11   caa        383 non-null    category
12   thall      383 non-null    category
13   output     383 non-null    category
dtypes: category(9), float64(5)
memory usage: 15.8 KB
```

Un cop assignades les columnes categòriques com a tals, comprovem que tenen el número esperat de classes (comparant amb l'imatge inicial):

```
In [6]: dict = {}
for i in cat_columns:
    dict[i] = df[i].value_counts().shape[0]
pd.DataFrame(dict, index=["unique count"]).transpose()

Out[6]:
```

	unique count
sex	2
cp	4
fbs	2
restecg	3
exng	2
slp	3
thall	4
caa	5
output	2

Tot sembla correcte en relació a la descripció inicial del dataset.

Passem ara als atributs numèrics. És important verificar que segueixen una distribució lògica (valors negatius quan no toquen, valors inconsistents, etc). Per a comprovar-ho, podem utilitzar el mètode .describe(), que ens mostra la distribució de cada una de les columnes numèriques:

```
In [7]: df.describe()

Out[7]:
```

	age	trtbps	chol	thalachh	oldpeak
count	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	131.623762	246.264026	149.646865	1.039604
std	9.082101	17.538143	51.830751	22.905161	1.161075
min	29.000000	94.000000	126.000000	71.000000	0.000000
25%	47.500000	120.000000	211.000000	133.500000	0.000000
50%	55.000000	130.000000	240.000000	153.000000	0.800000
75%	61.000000	140.000000	274.500000	166.000000	1.600000
max	77.000000	200.000000	564.000000	202.000000	6.200000

Totes les columnes numèriques semblen tenir valors consistents dins el que s'espera.

3.2 Identificació i gestió de valors extrems

Comprovem ara els outliers dels que presenta cada columna numèrica. Per a fer la representació, utilitzarem diagrames de caixa, que representa una forma molt simple i convenient per veure la distribució dels valors.

```
In [8]: flt_data = df[flt_columns]
fig, ax = plt.subplots()
ax.set_title('Outliers')
ax.boxplot(flt_data)
ax.set_xticklabels(flt_columns)
plt.show()

Outliers
```



Com podem veure gràcies al boxplot, les columna trtbps i chol són les que presenten valors més repartits. Tot i això, només destaquem un valor que realment sembla un outlier a la columna chol.

```
In [9]: df[df['chol'] >= 500]

Out[9]:
```

	age	sex	cp	trtbps	chol	fb	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
85	67.0	0	2	115.0	564.0	0	0	160.0	0	1.6	1	0	3	1

Al tractar-se d'un sol registre, el podem simplement eliminar, ja que podria afectar de manera negativa a estudis posteriors. Tot i això, en un cas real i depenent de la finalitat de l'estudi, caldria comprovar en detall que aquest valor és realment un outlier i no un valor que ens interessa tenir en compte.

```
In [10]: df = df.drop(df[df['chol'] >= 500].index)

In [11]: df[df['chol'] >= 500]

Out[11]:
```

	age	sex	cp	trtbps	chol	fb	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
--	-----	-----	----	--------	------	----	---------	----------	------	---------	-----	-----	-------	--------

Comprovem també si el dataset conté rows duplicades. Prenem la longitud del dataset abans i després de fer el drop_duplicates per veure si hi ha hagut algun canvi:

```
In [12]: length1 = len(df)
df.drop_duplicates(keep=False, inplace=True)
length2 = len(df)
print(f'El dataset inicial tenia {length1} files, després de treure duplicats té {length2} files.')
```

El dataset inicial tenia 382 files, després de treure duplicats té 380 files.

Hem eliminat doncs, dos registres repetits.

4. Anàlisi de les dades

4.1 Selecció dels grups de dades que es volen analitzar/comparar

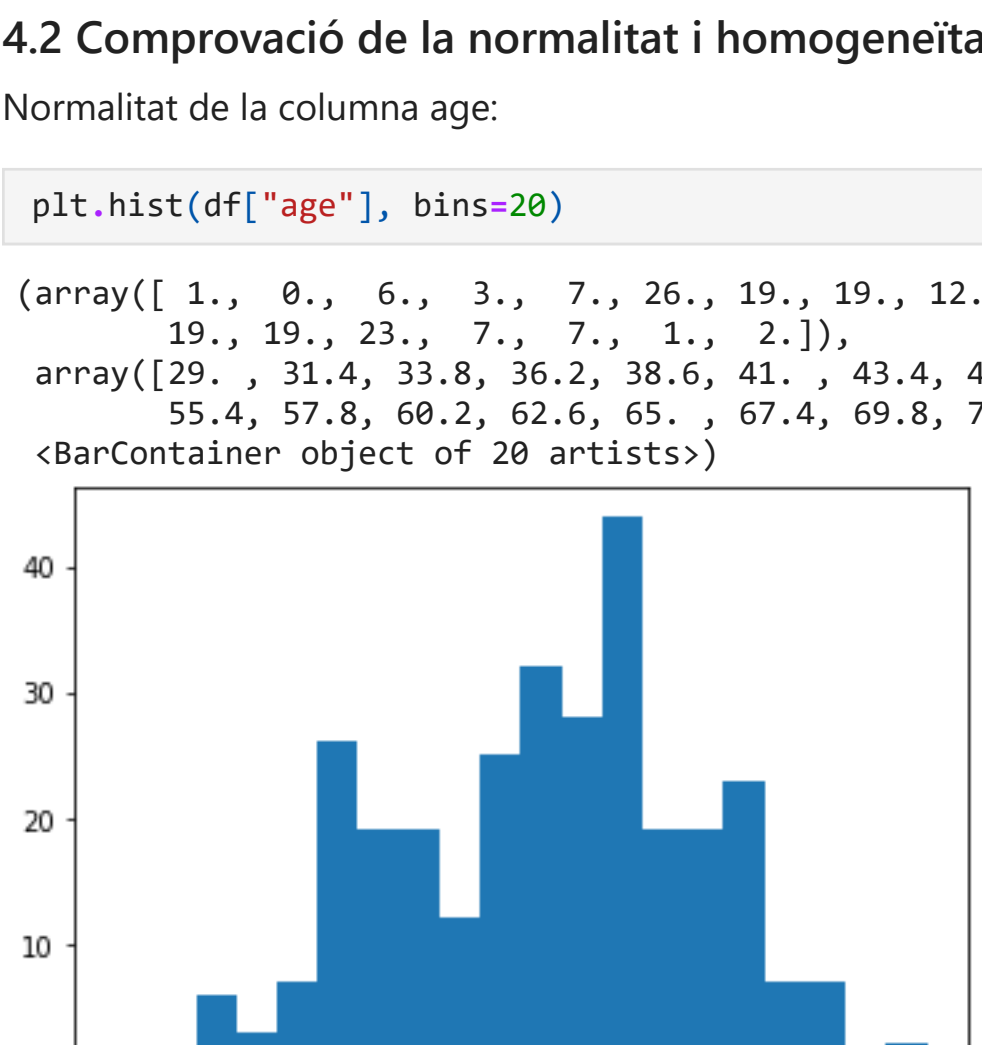
Compararem les columnes age/output i sex/output. També crearem un petit model de regressió logística amb la variable "thalachh".

4.2 Comprovació de la normalitat i homogeneïtat de la variància

Normalitat de la columna age:

```
In [13]: plt.hist(df["age"], bins=20)

Out[13]:
```



```
In [14]: stats.shapiro(df['age'])

Out[14]: ShapiroResult(statistic=-0.987129393577576, pvalue=0.008875136263668537)
```

Distribució de la columna sex:

```
In [15]: df["sex"].value_counts().plot(kind="pie", legend=True)

Out[15]:
```



Començem fent un test de Lavenne per evaluar si la nostra classe resultant (output) compleix la homogeneïtat de la variància:

```
In [16]: stat, p_value = stats.levene(df['age'], df['output'])

alpha = 0.05

print(f"El valor p és: {p_value}")

El valor p és: 1.9088423522591156e-81
```

4.3 Aplicació de proves estadístiques

Correlació Pearson entre variables numèriques:

```
In [17]: numeric_columns = list(df.select_dtypes(include=["int64", "float64"]).columns)
numeric_data = df[numeric_columns]
s_cn = numeric_data.corr(method="spearman") #multiple
fig, ax = plt.subplots(figsize=(8,6))
sb.heatmap(s_cn, cmap="Blues", annot=True)
plt.title('Pearson Correlation Matrix')
plt.show()

Pearson Correlation Matrix
```



Sexe - risc

La pregunta que volem respondre és: el sexe té alguna relació amb la probabilitat de tenir un atac cardíac?

Prova d'independència de chi-quadrat. Es tracta d'una prova estadística d'hipòtesi que s'utilitza per determinar si dues variables categòriques o nominals poden estar o no relacionades.

H0 = l'edat i el risc d'atac cardíac són variables independents
H1 = l'edat i el risc d'atac cardíac no són variables independents

Podem prendre com a coeficient $\alpha = 0.05$.

```
In [18]: crosstab = pd.crosstab(df["sex"], df["output"])

print(stats.chi2_contingency(crosstab)[1])

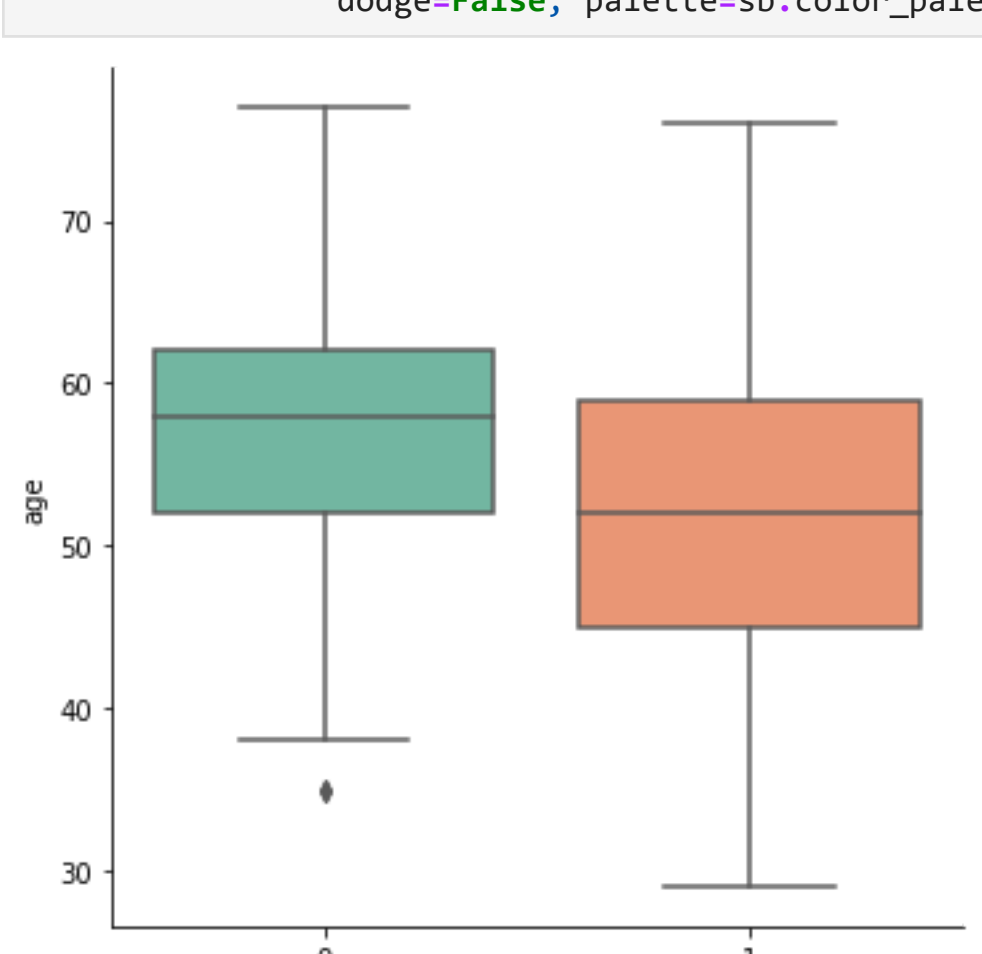
1.741525065120752e-06
```

Tenim un p-valor menor a 0.05, per tant rebutjem l'hipòtesi nul·la i concluem que el sexe pren importància a l'hora de saber si una persona presenta risc de tenir un atac cardíac.

Edat - risc

La pregunta que volem respondre és: l'edat té alguna relació amb la probabilitat de tenir un atac cardíac?

```
In [19]: #IV-DV relationships.
#Age and heart attack risk. Notably, individuals with a lower chance of
#heart attack are, on average, older than individuals with a high risk.
g = sb.catplot(kind="box", data=df, x="output", y="age", hue="output",
              dodge=False, palette=sb.color_palette("Set2"))
```



En aquest cas, simplement utilitzant un boxplot veiem que l'edat sí que influeix en el fet de tenir un atac cardíac, tal com suposavem. De totes maneres, no és una diferència molt significativa.

Creació d'un model de regressió logística

En aquest últim apartat crearem un model de regressió logística per veure si som capaços de predir el risc d'atac cardíac mitjançant alguna variable numèrica. Triem la variable "thalachh", que és el valor màxim de pulsació de cada individu. En un cas real i més complet, utilitzaríem totes (o la majoria) de les variables de les que disposem per la creació d'aquest model. Això, però, surt de l'abast d'aquesta pràctica.

```
In [20]: X = df.thalachh
y = df.output

In [21]: from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)

X_train = X_train.values.reshape(-1, 1)
X_test = X_test.values.reshape(-1, 1)
```

```
In [22]: # import the class
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(random_state=16)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
```

```
In [23]: from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

Out[23]:
```

	[24, 17],
	[6, 28], dtype=int64

```
In [24]: accuracy = metrics.accuracy_score(y_test, y_pred)
print(accuracy)

0.6933333333333334
```

Veïm que, tan sols utilitzant una variable, som capaços d'obtenir un 70% d'accuracy en el nostre dataset de test. Aquesta variable, doncs, serveix molt com a predictor del risc d'atac cardíac. Si en aquest model hi incorporéssim les altres variables de les que disposem, segurament obtindríem un model amb una precisió molt elevada.