Manejo y Análisis de Datos con Python

Guillermo Ruiz

Datos y extracción de información

¿Qué es la información? Este concepto ha sido muy usado en la sociedad moderna y ha tenido un papel fundamental para su desarrollo. Mantener una sociedad es imposible sin el **intercambio de información**. Aunque todos tenemos una buena idea de lo que representa esta palabra, es importante definirla de manera precisa, pero primero debemos hablar de datos.

Los datos se definen como símbolos que representan propiedades de objetos o fenómenos. Se obtienen de la observación y el intercambio de energía. Los datos son inútiles si no se representan de manera adecuada.

La información se extrae de los datos y se presenta en descripciones que responden preguntas como **quién, qué, cuándo y cuántos**. Los sistemas de información almacenan, ordenan, buscan y recuperan datos. La información se representa en un espacio menor que los datos.

El conocimiento es la generalización de la información para hacer predicciones. Requiere menor espacio que la información. Se puede adquirir de alguien que la tenga, de instrucciones o de la experiencia.

Finalmente, la sabiduría es la acumulación de conocimiento que permite su aplicación en nuevos problemas. Otorga la habilidad de ver lo que fue, es y será.

Figure 1: image.png

Gran cantidad de datos

En la actualidad, existe una fuente inagotable de datos: el internet. Además, existen cada vez más dispositivos que su función es recopilar, almacenar y procesar datos como las cámaras de vigilancia, equipos de monitoreo del clima o la contaminación, satélites, checadores digitales entre otros.

Toda esta información debe ser almacenada, procesada y analizada para convertirla en conocimiento.

Información estructurada y no estructurada

Esencialmente existen dos tipos de información, **la estructurada** y la **no estructurada**. Los datos estructurados tienen un formato fijo que facilita su lectura y su procesamiento. Por ejemplo, las bases de datos relacionales de un sistema de información

Por otro lado, a la información que no posee elementos como los de las bases de datos se le conoce como información no estructurada y puede tener formatos diferentes. Por ejemplo los documentos escritos, ya que en su interior pueden contener cualquier tipo de información.

Análisis de datos

- El análisis de los datos puede ser simple como calcular histogramas o más elaborados como técnicas de agrupamiento o clasificación de datos, encontrar correlaciones, etc.
- Cada objeto estudiado se describe mediante **variables** o atributos que representan sus propiedades.
- Todas las variables de un objeto se llama **registro**.
- A la colección de todos los registros se le conoce como **conjunto de datos**, que generalmente se representa por una tabla donde cada fila es un registro y las columnas son los atributos.

Existen diferentes tipos de variables, según el tipo de datos que contienen. Los tipos más comunes son: - Variable nominal. Indica una categoría a la que pertenece el objeto. La categoría pueden ser palabras o números. Sus elementos no tienen un orden. - Variable binaria. Toma los valores de 0 ó 1. - Variables Ordinales. Los valores que toma tienen un orden. - Variables de intervalo. Son variables numéricas donde cada número representa un intervalo.

Aprendizaje Supervisado

- En el aprendizaje supervisado se tiene los datos y cada uno de ellos tiene asociado un valor objetivo.
- La **clasificación** es uno de los problemas más comunes en el aprendizaje supervisado. En él, se tiene que asignar la etiqueta correcta a los datos.
- Los algoritmos de aprendizaje se alimentan de una gran colección de datos que ya fueron previamente etiquetados por expertos. Los algoritmos buscan patrones que les permitan clasificar de manera correcta datos nunca antes vistos ellos. A este proceso se le llama **entrenamiento**.

Aprendizaje no supervisado

- En el aprendizaje no supervisado, se tienen los datos como en el caso anterior, pero no tienen asociado ningún valor.
- El objetivo entonces es agrupar los datos parecidos y diferenciarlos del resto. A este proceso se le llama **clustering**.
- Los clusters formarán una partición de los datos y cada cluster es un conjunto que contiene objetos similares entre ellos pero distintos a los que hay en los demás clusters.

- Por ejemplo, tenemos un conjunto de datos que contiene artículos de noticias provenientes de diarios nacionales. Sólo tenemos los textos pero no están etiquetados o separados por tema. El objetivo es crear un algoritmo que sea capaz de separarlos por tema, que ponga todos los de política en un conjunto, en otro los de deportes, y así para los temas de finanzas, internacionales y salud.

Repaso

Vamos a trabajar en Colab que es un servicio de Google donde tenemos un **Notebook** y tenemos acceso a una máquina virtual donde se va a ejecutar nuestro código.

Notebook

Se puede poner texto en **negrita** o *cursiva*

```
print("hola mundo")
```

Para más información ver este enlace

```
print("hola mundo")
```

```
hola mundo
```

```
x = 0
x
```

```
0
```

```
x = x + 1
x
```

```
1
```

Listas

Las listas son una estructura de datos básica donde los elementos tienen un orden y puede haber repetidos.

```
l = [1, 2, 3, 4, 5, 6]
```

```
l[0]
```

```
1
```

```
l[3]
```

```
4
```

```
l[2:5]
```

```
[3, 4, 5]
```

```
l[:4]
```

```
[1, 2, 3, 4]
```

```
l[3:]
```

```
[4, 5, 6]
```

```
l[-1]
```

```
6
```

```
for v in l:
  print(v*2)
```

```
2
4
6
8
10
12
```

```
len(l)
```

```
6
```

```
l.append(7)
l
```

```
[1, 2, 3, 4, 5, 6, 7]
```

```
l2 = [4,3,6,2,1,7,5]
l2.sort()
l2
```

```
[1, 2, 3, 4, 5, 6, 7]
```

Pandas

Pandas es una librería para manejar datos tabulares. Puede leer archivos csv, json, xls entre muchos otros.

```
import pandas as pd
```

```
from sklearn.datasets import fetch_california_housing
california_housing = fetch_california_housing(as_frame=True)
df = california_housing.data
```

```
print(california_housing.DESCR)
```

```
.. _california_housing_dataset:

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 group
    - HouseAge      median house age in block group
    - AveRooms      average number of rooms per household
    - AveBedrms     average number of bedrooms per household
    - Population    block group population
    - AveOccup      average number of household members
    - Latitude      block group latitude
    - Longitude     block group longitude

:Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts,
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census
block group. A block group is the smallest geographical unit for which the U.S.
Census Bureau publishes sample data (a block group typically has a population
of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average
number of rooms and bedrooms in this dataset are provided per household, these
columns may take surprisingly large values for block groups with few households
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

.. rubric:: References

- Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,
  Statistics and Probability Letters, 33 (1997) 291-297
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 8 columns):
```

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
dtypes: float64(8)
memory usage: 1.3 MB
```

```
df
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

```
df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

```
df.shape
```

```
NameError: name 'df' is not defined
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 df.shape
NameError: name 'df' is not defined
```

El método `describe` nos dice un resumen de las estadísticas de los datos que tiene.

```
df.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.00
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.63186
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.54000
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.93000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.26000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.71000
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.95000

Se puede seleccionar una o varias columnas de los datos.

```
df['HouseAge']
```

```
0        41.0
1        21.0
2        52.0
3        52.0
4        52.0
        ...
20635    25.0
20636    18.0
20637    17.0
20638    18.0
20639    16.0
Name: HouseAge, Length: 20640, dtype: float64
```

```
sub = df[['HouseAge', 'Latitude', 'Longitude']]
sub
```

	HouseAge	Latitude	Longitude
0	41.0	37.88	-122.23
1	21.0	37.86	-122.22
2	52.0	37.85	-122.24
3	52.0	37.85	-122.25
4	52.0	37.85	-122.25
...	...	...	...
20635	25.0	39.48	-121.09
20636	18.0	39.49	-121.21
20637	17.0	39.43	-121.22
20638	18.0	39.43	-121.32
20639	16.0	39.37	-121.24

```
sub.head()
```

	HouseAge	Latitude	Longitude
0	41.0	37.88	-122.23
1	21.0	37.86	-122.22
2	52.0	37.85	-122.24
3	52.0	37.85	-122.25
4	52.0	37.85	-122.25

También se puede seleccionar un subconjunto de los datos basada en una condición. En este caso la condición es `df['HouseAge']<10`. También se puede usa `==`, `!=`, `<`, `<=` para obtener la condición.

```
df[df['HouseAge']<10]
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
59	2.5625	2.0	2.771930	0.754386	94.0	1.649123	37.82	-122.29
570	7.6110	5.0	6.855776	1.061442	7427.0	2.732524	37.72	-122.24
577	7.0568	5.0	7.023438	0.912109	1738.0	3.394531	37.73	-122.06
631	3.6630	5.0	5.158537	1.213415	814.0	2.481707	37.72	-122.17
676	5.4858	5.0	5.204489	1.014963	840.0	2.094763	37.68	-122.18

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
...	...	...	...	...	...	...	...	...
20563	3.1250	9.0	5.148007	1.119593	6837.0	2.899491	38.67	-121.75
20570	3.2222	6.0	4.821429	1.032468	894.0	2.902597	38.54	-121.96
20571	3.4186	6.0	5.662562	1.027094	2561.0	3.153941	38.53	-121.99
20588	2.0474	5.0	4.378132	1.113895	938.0	2.136674	39.15	-121.59
20627	3.0000	5.0	6.067797	1.101695	169.0	2.864407	39.13	-121.32

```
df[df['HouseAge'].isin([2,4,6])]
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
59	2.5625	2.0	2.771930	0.754386	94.0	1.649123	37.82	-122.29
702	3.8958	6.0	4.267016	1.029668	1189.0	2.075044	37.63	-122.02
838	5.0406	6.0	6.016166	1.094688	1568.0	3.621247	37.61	-122.08
854	2.8750	2.0	5.624161	1.979866	240.0	1.610738	37.59	-122.01
863	5.8151	6.0	6.402616	1.042151	2071.0	3.010174	37.58	-122.00
...	...	...	...	...	...	...	...	...
20445	5.4493	6.0	5.772688	1.006050	3196.0	2.762316	34.33	-118.83
20450	5.2206	6.0	5.749529	1.067797	3246.0	3.056497	34.28	-118.91
20457	4.5458	6.0	5.154015	1.061606	2180.0	2.398240	34.28	-118.77
20570	3.2222	6.0	4.821429	1.032468	894.0	2.902597	38.54	-121.96
20571	3.4186	6.0	5.662562	1.027094	2561.0	3.153941	38.53	-121.99

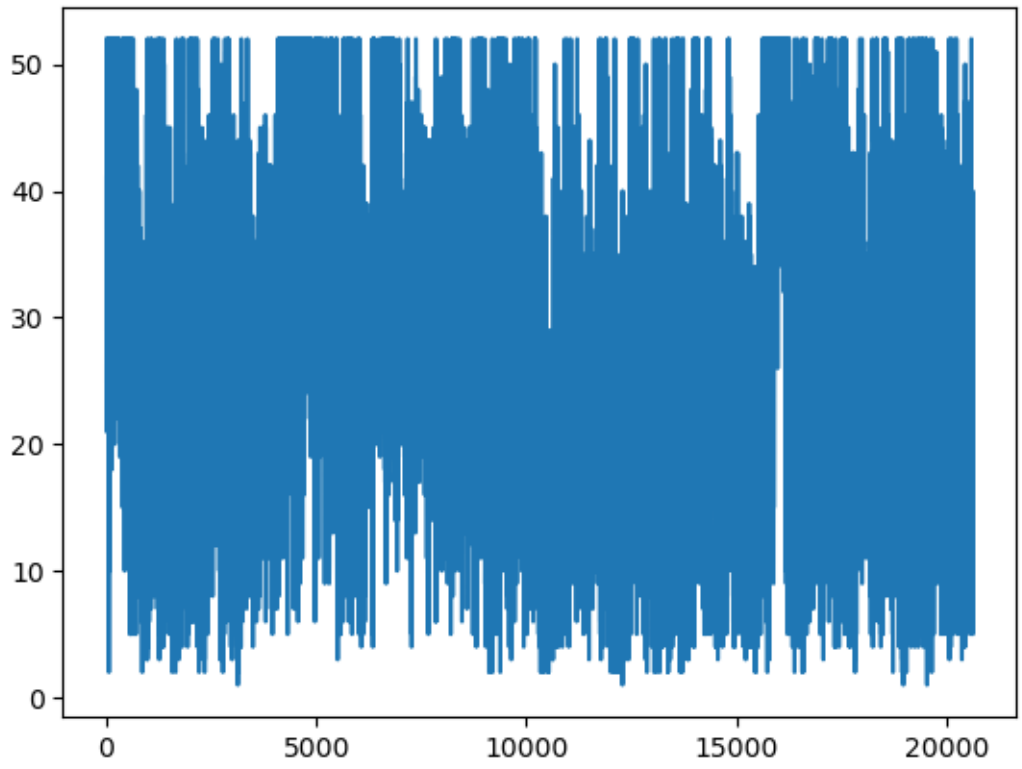
```
df[(df['HouseAge']<10) | (df['HouseAge']>50)]
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
5	4.0368	52.0	4.761658	1.103627	413.0	2.139896	37.85	-122.25
6	3.6591	52.0	4.931907	0.951362	1094.0	2.128405	37.84	-122.25
...	...	...	...	...	...	...	...	...
20570	3.2222	6.0	4.821429	1.032468	894.0	2.902597	38.54	-121.96
20571	3.4186	6.0	5.662562	1.027094	2561.0	3.153941	38.53	-121.99
20588	2.0474	5.0	4.378132	1.113895	938.0	2.136674	39.15	-121.59
20592	0.8928	52.0	4.442953	1.073826	520.0	3.489933	39.14	-121.58
20627	3.0000	5.0	6.067797	1.101695	169.0	2.864407	39.13	-121.32

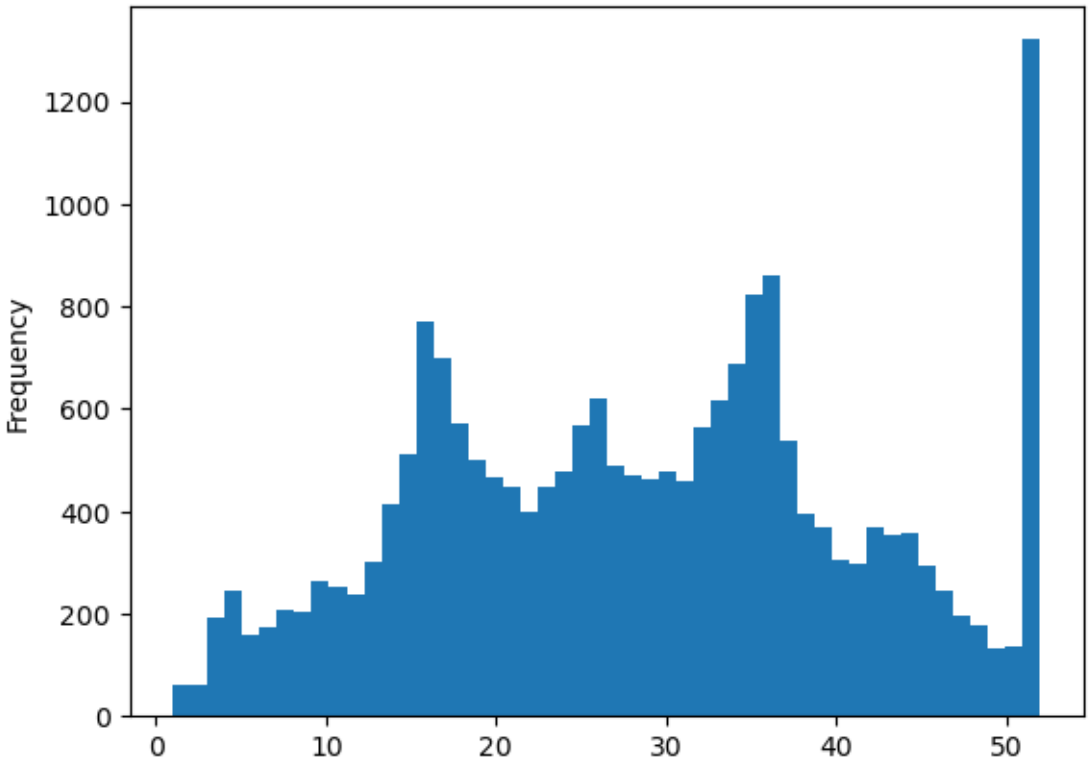
Graficar

```
import matplotlib.pyplot as plt
```

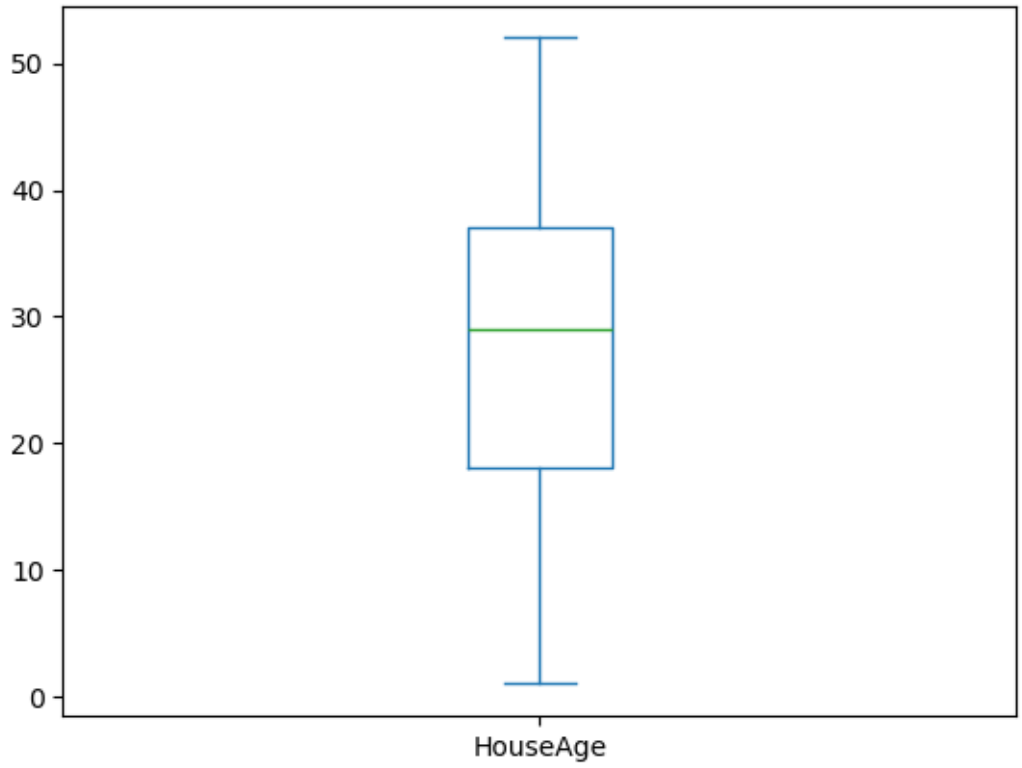
```
df['HouseAge'].plot()
plt.show()
```



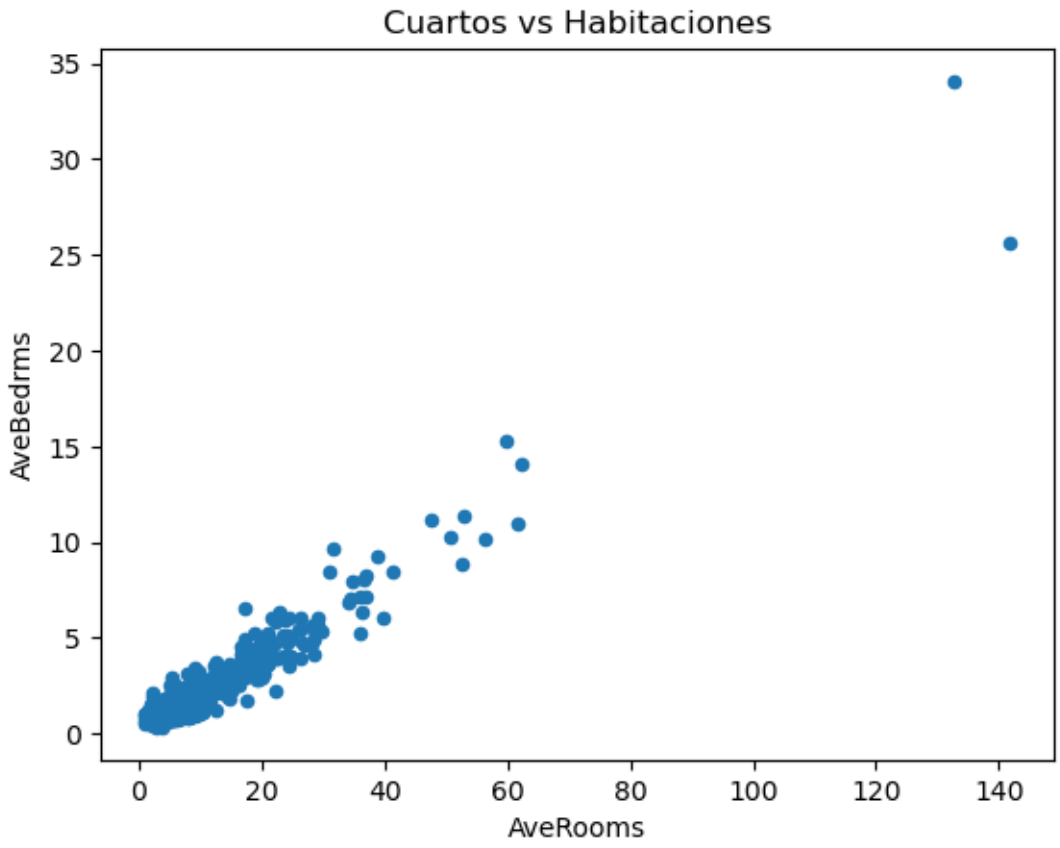

```
df['HouseAge'].plot.hist(bins=50)
plt.show()
```




```
df['HouseAge'].plot.box()
plt.show()
```




```
df.plot.scatter(x='AveRooms', y='AveBedrms')
plt.title('Cuartos vs Habitaciones')
plt.show()
```



Ejercicio

El conjunto de datos **heart** incluye 13 características y una variable objetivo (target). Las varacterísticas son:

1. age
2. sex
3. chest pain type (4 values)
4. resting blood pressure
5. serum cholestoral in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment

12. number of major vessels (0-3) colored by flourosopy
13. thal: 0 = normal; 1 = fixed defect; 2 = reversable defect
14. **target**: 0 = no disease; 1 = disease

Para este ejercicios se debe hacer los siguiente:

1. Leer el conjunto de datos con Pandas.
2. Mostrar los registros de personas con más de 70 años.
3. Crear dos DataFrames, uno para las personas enfermas y otro para las sanas.
4. Mostrar el histograma de edad para las personas enfermas y otro para las sanas.
5. Graficar los valores de las variables **age** contra **trestbps**

```
heart = pd.read_csv("heart.csv")
heart.head(3)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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