# Análisis Exploratorio de Datos con Pandas en Python

## ¿Qué es el EDA?

EDA es la sigla en inglés para Exploratory Data Analysis y consiste en una de las primeras tareas que tiene que desempeñar el Científico de Datos. Es cuando revisamos por primera vez los datos que nos llegan, por ejemplo, un archivo CSV que nos entregan y deberemos intentar comprender “¿de qué se trata?”, **vislumbrar posibles patrones y reconociendo distribuciones estadísticas que puedan ser útiles en el futuro.**

OJO!, lo ideal es que **tengamos un objetivo** que nos hayan “adjuntado” con los datos, que indique lo que se quiere conseguir a partir de esos datos. Por ejemplo, nos pasan un excel y nos dicen “Queremos [*predecir ventas*](https://www.aprendemachinelearning.com/pronostico-de-ventas-redes-neuronales-python-embeddings/) a 30 días”(Redes Neuronales), ó “[*Clasificar*](https://www.aprendemachinelearning.com/arbol-de-decision-en-python-clasificacion-y-prediccion/) casos malignos/benignos de una enfermedad” (Arbol de decisión), “Queremos [*identificar audiencias*](https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/) que van a realizar re-compra de un producto” (K-Means), “queremos hacer [*pronóstico*](https://www.aprendemachinelearning.com/regresion-logistica-con-python-paso-a-paso/) de fidelización de clientes/abandonos” (Regresión logistica), “Quiero [*detectar casos de fraude*](https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/) en mi sistema en tiempo real” (Clasificación de datos desbalanceados).

## EDA deconstruido

Al llegar un archivo, lo primero que deberíamos hacer es intentar responder a las siguientes preguntas:

* ¿Cuántos registros hay?
  + ¿Son demasiado pocos?
  + ¿Son muchos y no tenemos Capacidad (CPU+RAM) suficiente para procesarlo?
* ¿Están todas las filas completas o tenemos campos con valores nulos?
  + En caso que haya demasiados nulos: ¿Queda el resto de información inútil?
* ¿Qué datos son discretos y cuáles continuos?
  + Muchas veces sirve obtener el tipo de datos: texto, int, double, float
* Si es un problema de tipo supervisado:
  + ¿Cuál es la columna de “salida”? ¿binaria, multiclase?
  + ¿Está balanceado el conjunto salida?
* ¿Cuáles parecen ser features importantes? ¿Cuáles podemos descartar?
* ¿Siguen alguna distribución?
* ¿Hay correlación entre features (características)?
* En [problemas de NLP](https://www.aprendemachinelearning.com/procesamiento-del-lenguaje-natural-nlp/) es frecuente que existan categorías repetidas o mal tipeadas, o con mayúsculas/minúsculas, singular y plural, por ejemplo “Abogado” y “Abogadas”, “avogado” pertenecerían todos a un mismo conjunto.
* ¿Estamos ante un problema dependiente del tiempo? Es decir un [TimeSeries](https://www.aprendemachinelearning.com/pronostico-de-series-temporales-con-redes-neuronales-en-python/).
* Si fuera un problema de [Visión Artificial](https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/) (Redes Neuronales Convolucionales): ¿Tenemos suficientes muestras de cada clase y variedad, para poder hacer generalizar un modelo de Machine Learning?
* ¿[Cúales son los Outliers](https://www.aprendemachinelearning.com/deteccion-de-outliers-en-python-anomalia/)? (unos pocos datos aislados que difieren drásticamente del resto y “contaminan” o desvían las distribuciones)
  + ¿Podemos eliminarlos? ¿es importante conservarlos?
  + ¿Son errores de carga o son reales?
* ¿Tenemos posible sesgo de datos? (por ejemplo perjudicar a [clases minoritarias](https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/) por no incluirlas y que el modelo de ML discrimine)

Puede ocurrir que tengamos set de datos incompletos y debamos pedir a nuestro cliente/proveedor o interesado que nos brinde mayor información de los campos, que aporte más conocimiento o que corrija campos. [¿Qué son los conjuntos de Train, Test y Validación en Machine Learning?](https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/)

También puede que nos pasen múltiples fuentes de datos, por ejemplo, un csv, un excel y el acceso a una base de datos. Entonces tendremos que hacer un paso previo de unificación de datos.

## ¿Qué sacamos del EDA?

El EDA será entonces una primera aproximación a los datos, si estamos más o menos bien preparados y suponiendo una muestra de datos “suficiente”, puede que en “unas horas” tengamos ya varias conclusiones como, por ejemplo:

* ¡Esto que quiere hacer el cliente CON ESTOS DATOS es una locura imposible! (esto ocurre la mayoría de las veces jeje)
* No tenemos datos suficientes o son de muy mala calidad, pedir más al cliente.
* Un [modelo de tipo Árbol](https://www.aprendemachinelearning.com/arbol-de-decision-en-python-clasificacion-y-prediccion/) es lo más recomendado usar (¡reemplazar Árbol, por el tipo de modelo que hayamos descubierto como mejor opción!)
* No hace falta usar Machine Learning para resolver lo que pide el cliente. (¡ESTO ES MUY IMPORTANTE!)
* Es todo tan aleatorio que no habrá manera de detectar patrones
* Hay datos suficientes y de buena calidad como para seguir a la próxima etapa.

A estas alturas podemos saber si nos están pidiendo algo viable o si necesitamos más datos para comenzar.

**Repito por si no quedó claro: el EDA debe tomar horas, o puede que un día, pero la idea es poder sacar algunas conclusiones rápidas para contestar al cliente si podemos seguir o no con su propuesta.**

Después del EDA, suponiendo que seguimos adelante podemos tomarnos más tiempo y analizar en mayor detalle los datos y avanzar a nuevas etapas para aplicar modelos de Machine Learning.

## Técnicas para EDA

¡Vamos a lo práctico!, ¿Qué herramientas tenemos hoy en día? La verdad es que **como cada conjunto de datos suele ser único, el EDA se hace bastante “a mano”**, pero podemos seguir diversos pasos ordenados para intentar acercarnos a ese objetivo que nos pasa el cliente en pocas horas.



1. **Realizar un análisis descriptivo de las variables**, para obtener una idea representativa del conjunto de datos.
2. **Re-ajustar los tipos de las variables** para que sean consistentes en el momento de realizar posteriores operaciones.
3. **Detección y tratamiento de datos ausentes**. El tratamiento o la eliminación de datos ausentes es esencial, ya que de otra manera no será posible procesar adecuadamente las variables numéricas.
4. **Identificación de datos atípicos y su tratamiento,** dado que pueden distorsionar futuros análisis estadísticos.
5. **Realizar un examen numérico y gráfico de las relaciones entre las variables analizadas para determinar el grado de correlación entre ellas**, pudiendo predecir el comportamiento de una variable en función de las otras.



Veamos a continuación de forma detallada cada una de las etapas propuestas para llevar a cabo un análisis exploratorio de datos. Cada capítulo incluye la sección **"Experimenta”** que, por medio de la aplicación práctica de diversas funciones en python, te ayudará a comprender los conceptos que se explican.

## Análisis Descriptivo



Una vez que se ha obtenido el dataset sobre [el registro de la calidad del aire en la Comunidad Autónoma de Castilla y León](https://datos.gob.es/es/catalogo) del catálogo de datos abiertos y tenemos cargados los datos en nuestro entorno de desarrollo para posteriormente llevar a cabo alguna tarea de reutilización de los mismos como, por ejemplo, una visualización interactiva o el desarrollo de una aplicación, es recomendable obtener una vista descriptiva sobre el contenido de las tablas de datos con las que vas a trabajar. Con este fin aplicaremos funciones de estadística descriptiva para explorar la estructura del conjunto de datos y examinar los datos y variables que presenta. Asimismo, será muy útil el uso de determinadas representaciones graficas que te ayudarán a intuir la forma que poseen las distribuciones de los datos.

Experimenta

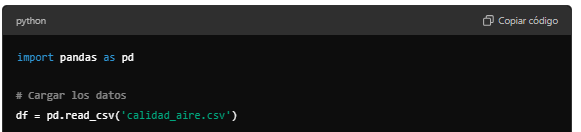
La tabla de datos cargada en nuestro entorno de desarrollo, con la que trabajaremos desde este momento y a la cual aplicaremos todos los procesos asociados al análisis exploratorio de datos propuestos en esta guía, se denomina calidad\_aire. Esta tabla contiene los datos de la calidad del aire de la Comunidad Autónoma de Castilla y León.

Para esta tarea, usaremos funciones de Python de la librería pandas que nos mostrarán una visión general de la tabla de datos. Además de en esta etapa inicial, se utilizarán a lo largo del EDA con el fin de observar los cambios que progresivamente se irán efectuando sobre los datos a medida que realizamos las diferentes tareas de análisis exploratorio.

La función head() muestra el contenido de las 5 primeras filas de la tabla de datos que acabamos de cargar; la función info() permite conocer de forma compacta la estructura interna de la tabla de datos, indicando el tipo de variables, los rangos de valores y una muestra de dichos valores. De un vistazo, se puede apreciar la presencia de valores ausentes identificados mediante NaN (Not a Number). La función describe() muestra un resumen general de las variables de la tabla, incluyendo los valores mínimo, máximo, media, mediana, primer y tercer cuartil para las variables numéricas, además del número específico de valores NaN presentes en cada una.

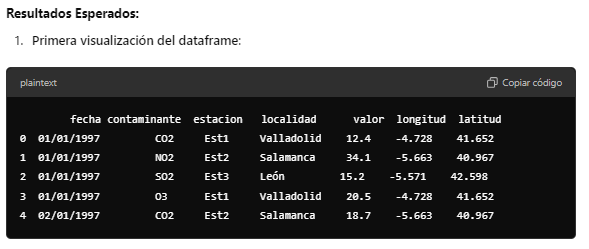
Veamos una muestra de la información descriptiva de las funciones mencionadas:

**1. Cargar el Dataset:** Primero, asegurémonos de que tenemos los datos en un DataFrame de pandas. Puedes cargar el dataset desde un archivo CSV (o cualquier otro formato)

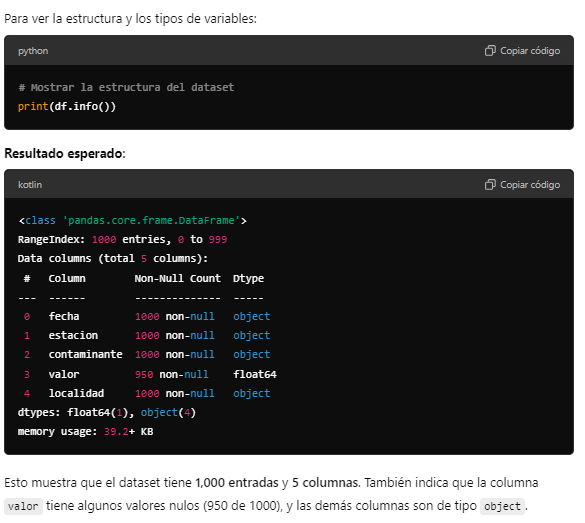


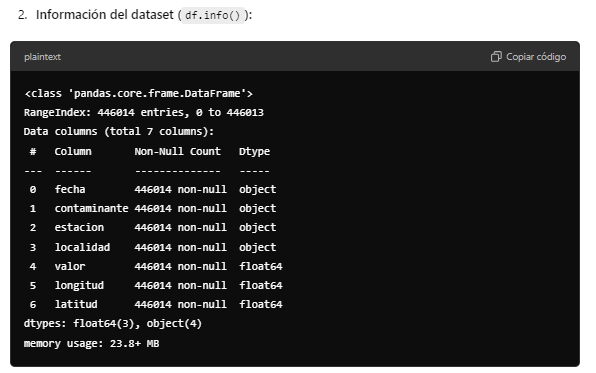
**2. Visualizar el Contenido de la Tabla de Datos:** Para obtener una vista general del contenido del DataFrame, utilizamos la función *head*() de *pandas*:



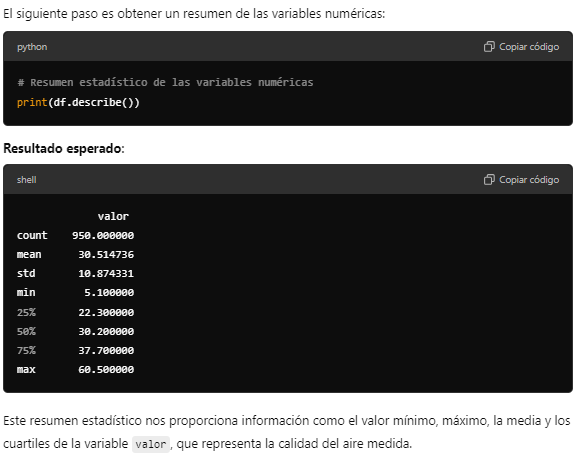


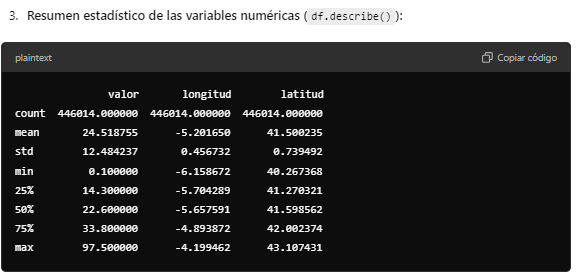
**3. Estructura Interna del DataFrame:** Para conocer la estructura interna del DataFrame, es decir, los tipos de variables y la presencia de valores ausentes, utilizamos *info*()

****

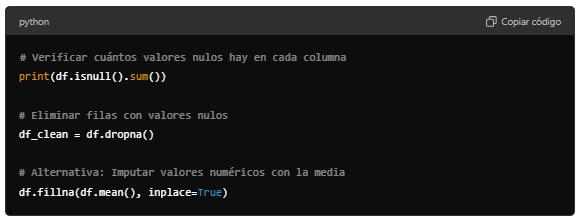
****

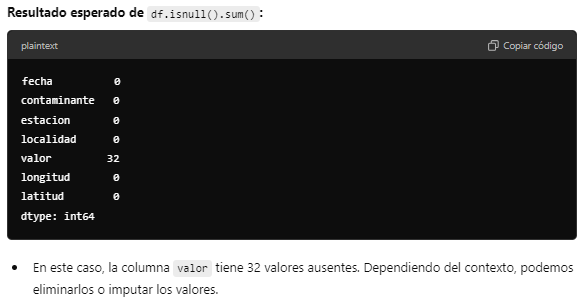
**4. Resumen Estadístico de las Variables: Para obtener un resumen estadístico que incluya el** mínimo, máximo, media, mediana, primer y tercer cuartil, así como el conteo de valores ausentes, usamos *describe*()





**5. Detección y Tratamiento de Valores Ausentes:** Puedes identificar los valores ausentes en cada columna con





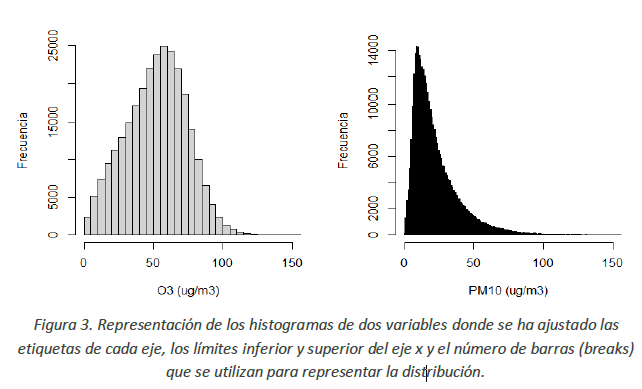
Según los resultados obtenidos, las **principales características** que presenta la tabla de datos son:

* El número total de observaciones del conjunto de datos es de 446.014.
* **El número total de variables es 7**, cuatro de tipo Object y 3 numéricas.
* El rango temporal abarca desde el 01/01/1997 hasta el 31/12/2020**.**
* **Presenta dos variables del Sistema de Coordenadas Geográficas**: Longitud y Latitud, para georreferenciar cada una de las estaciones de calidad de aire de Castilla y León.

**Es importante acompañar esta tarea con la representación de gráficos:** histogramas, gráficos de líneas, barras o sectores, entre otros, para observar el comportamiento de la distribución de los datos. Una de las representaciones más útiles es la que se obtiene aplicando la función *hist*() de *matplotlib* o *seaborn*, que permite generar un histograma para observar la distribución de cualquier variable numérica presente en el conjunto de datos. Generamos los histogramas de dos variables numéricas que presenta la tabla de datos: 03 (μg/m3) y PM10 ((μg/m3)



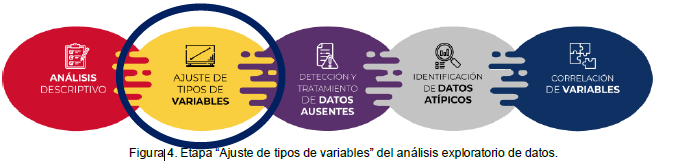
El código generará dos histogramas en una sola figura: uno para O3 y otro para PM10. Cada histograma mostrará la frecuencia de las mediciones en el rango especificado, permitiéndote visualizar la distribución de cada contaminante en el conjunto de datos.



Observando los histogramas de ambas variables, podemos concluir que presentan una distribución sesgada hacia la derecha, con valores más cercanos al 0, aunque este sesgo es mucho más pronunciado en la variable **PM10.** Este resultado debe tenerse en cuenta en futuros análisis estadísticos, ya que las variables deben tratarse de manera distinta en función de la distribución que presenten sus datos.

Como hemos indicado anteriormente, **es aconsejable realizar un análisis descriptivo cada vez que se modifique la tabla de datos para comprobar de esta manera el efecto que producen sobre los datos los cambios aplicados en cada nueva etapa del EDA**.

## Ajuste de los tipos de variables



Una de las primeras comprobaciones que hay que hacer tras cargar los datos en el entorno de trabajo, es **verificar que cada variable se ha almacenado con el tipo de valor que corresponde**. Por ejemplo, que las variables que contienen valores numéricos representan números y las cualitativas o categóricas están tipificadas como cadenas de caracteres y contienen una cantidad finita de elementos.

**Los tipos de variables** habituales, que puede albergar nuestra tabla de datos pueden ser:

* **numérico**, almacena números que pueden ser decimales o enteros.
* **carácter**: alberga cadenas de texto.
* **categórico**: contiene un número limitado de valores o categorías de información.
* **lógico o booleano**: variables binarias que solo pueden tomar dos valores: TRUE y FALSE o 0 y 1; pueden ser resultado de una comparación o condición de otras variables presentes en el conjunto de datos.
* **fecha**: almacena intervalos específicos de tiempo.

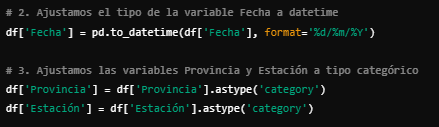
Experimenta

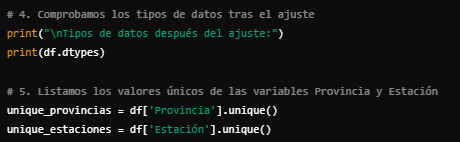
Con la función *info*(), vista en el bloque anterior, podemos descubrir el tipo de cada una de las variables presentes en nuestro conjunto de datos, así como un ejemplo de los valores que toma.

En nuestro conjunto de datos, a priori, encontramos varias variables donde el tipo de dato no se corresponde con la naturaleza del valor que contiene, como es el caso de la variable Fecha, Provincia y Estación. Estas tres variables han sido codificadas como tipo object, sin embargo:

* La variable **Fecha**, debería ser de tipo fecha ya que almacena la fecha en la que se recogieron los datos para el resto de variables y de esta forma será posible aplicar determinadas funciones predefinidas en Python para trabajar con datos de este tipo. Por ejemplo, obtener la diferencia en días entre dos periodos de tiempo. Para transformar una cadena de caracteres a una variable fecha usaremos la función de la librería pandas *pd.to\_datetime(df[‘nombre\_columna’], format=’’),* asegurando previamente si la secuencia correcta de día (d), mes (m) y año (y) o se utiliza otro formato internacional de fecha.
* Las variables **Provincia** y **Estación**, son variables de tipo categórico. Para transformar una cadena de caracteres a una variable tipo categórico, usaremos la función *df[‘nombreColumns’].astype(‘category’),* conociendo previamente la lista de valores que puede tomar la variable. Para conocer la lista de valores que toma la variable y comprobar si finalmente esa variable toma un número finito de valores usaremos la función *unique*(), que devuelve un vector con los valores que presenta la variable sin duplicados.

El procedimiento a seguir es re-ajustar los tipos de estas variables para poder realizar posteriormente las operaciones, análisis y representaciones gráficas que sean necesarias.





## Detección y tratamiento de datos ausentes



## Detección de datos ausentes

**La presencia de datos ausentes, perdidos, missing values, o celdas vacías,** representados habitualmente en Python como NaN (Not a Number), **es una problemática habitual en muchos conjuntos de datos**. La mayoría de las veces se debe a fallos en la transcripción de los datos o problemas durante la recogida de datos, por ejemplo, debido a la imposibilidad de obtener cierta medida u observación.

**Tratar con conjuntos de datos en los que existen datos ausentes puede generar problemas a la hora de aplicar diferentes análisis estadísticos o en la generación de representaciones gráficas.** A fin de evitar problemas futuros, es necesario aprender a detectar y aplicar algún tipo de tratamiento.

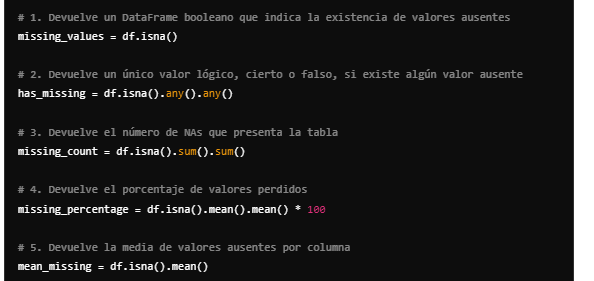
Experimenta

Con la función isna() de pandas, comprobamos la existencia de valores ausentes. El resultado de esta función es un DataFrame booleano, donde cada celda es True si el valor es NaN y False en caso contrario.

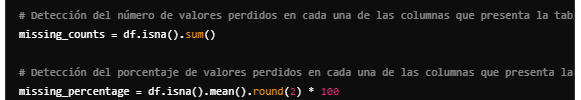
Existen otras funciones muy útiles que también se pueden usar para evaluar la existencia de NaN:

* La función any() combinada con isna() devuelve True si el DataFrame presenta al menos un valor ausente, sin indicar el número de valores perdidos ni su posición.
* La función sum() combinada con isna() permite determinar el número total de valores ausentes.
* La función mean() combinada con isna() muestra el porcentaje de valores perdidos en el DataFrame.

Con la función *isna*() de pandas, podemos comprobar la existencia de valores ausentes. El resultado de esta función es un DataFrame booleano del mismo tamaño que el original, donde cada celda devuelve True si hay un valor ausente (NaN) y False en caso contrario.



En ocasiones, conviene realizar una detección de valores ausentes por columnas, en lugar de por filas, para identificar si alguna de las variables del dataset presenta un determinado nivel de datos perdidos. Para ello, podemos utilizar las funciones *sum()* y *mean()* junto con *isna()* en pandas.



Analizando el conjunto de datos con el cual estamos trabajando en esta guía, podemos observar que la tabla “**calidad\_aire**” presenta un total de 116.281 valores perdidos, el 21% del total. Si analizamos los NANs, en cada una de las variables, dos de ellas: **CO (mg/m3)** y **PM25 (μg/m3),** presenta un porcentaje superior al 50%, del 77% y 88% respectivamente, lo que conlleva una ausencia significativa de información en el dataset de trabajo. **Esta anomalía debe ser tratada de alguna forma para disminuir su impacto en el objetivo de reutilización de los datos.**

## Tratamiento de datos ausentes

Existen varias **maneras de tratar con valores ausentes**:

* Rellenar los valores con la media, mediana o el valor más frecuente de la variable.
* Completar los valores que faltan por el valor que esté directamente antes o después en la fila o columna.
* Completar todos los datos faltantes con 0, si se trata de valores numéricos. Esta opción es poco aconsejable ya que puedes modificar de manera significativa los resultados.
* Eliminar las filas que presenten valores ausentes, siempre y cuando el conjunto de datos sea lo suficientemente grande y no se pierde información relevante al eliminar esas filas.
* Y una forma abrupta de tratamiento que depende del contexto de análisis, es eliminar las variables que presentan un porcentaje mayor del 50% de datos ausentes.

**Seleccionar la mejor manera de abordar el tratamiento de valores ausentes, depende del tipo de dato, del tratamiento posterior de los mismos o de la causa de la falta de esos valores** (si se conoce). La estrategia más común es utilizar el valor medio, pero, que sea la opción más popular, no significa necesariamente que sea la elección correcta para el conjunto de datos con el cual estemos tratando en ese momento.

**Los tratamientos de datos ausentes mencionados pueden modificar los resultados obtenidos en futuros análisis, disminuir el tamaño muestral o introducir un sesgo**, por lo que, un diseño riguroso del EDA implica documentar esta decisión con el objetivo de mantener trazabilidad de los procesos llevados a cabo y poder en todo momento, retornar a uno de estos puntos de decisión ante una determinada inconsistencia o debilidad en etapas posteriores del análisis de datos.

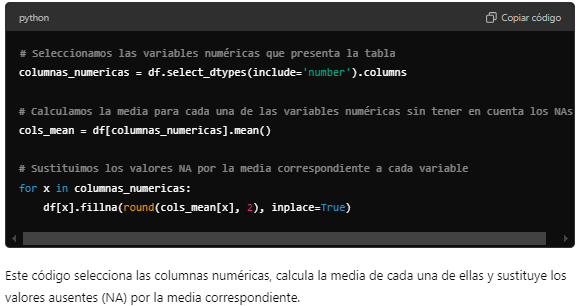
Experimenta  
Como ejemplo de aplicación de las opciones enumeradas, el primer tratamiento que vamos a realizar sobre los datos perdidos es la eliminación de las columnas que presentan un porcentaje superior al 50%, ya que un número de NaNs tan alto puede producir errores o distorsionar los análisis posteriores al no ser usadas las filas que presentan NaNs (en este caso, no se usaría más del 50% de las observaciones).



**Filtrado de Columnas:**

* df.loc[:, df.isna().mean() < 0.50] selecciona solo las columnas cuyo porcentaje de NaNs es menor al 50%.

Continuando con el ejemplo, los valores perdidos que presenta la tabla en el resto de variables, los sustituiremos por la media de cada una de las columnas, para no perder información significativa y los análisis posteriores no se vean alterados.



Las operaciones realizadas ilustran las posibilidades para el tratamiento de datos ausentes sin entrar a valorar la oportunidad de su aplicación respecto a un análisis riguroso de la calidad del aire dado que en ese caso influyen otros factores meteorológicos que en este ejemplo no se están teniendo en cuenta.

## Detección y tratamiento de valores atípicos (outliers)



**Un valor atípico u *outlier*, es una observación significativamente distinta del resto de datos que presenta una variable**, de tal magnitud que se puede considerar un valor anómalo. **Estos valores pueden afectar a tareas siguientes pudiendo llegar a modificar los resultados**. Es necesario detectarlos y tratarlos para poder disminuir su influencia en los análisis posteriores o, en casos muy extremos, eliminarlos del conjunto de datos.

Lo más recomendable para el tratamiento de los datos atípicos es reducir su posible influencia en los análisis. Aunque no es objeto de esta guía, hay que mencionar que existen métodos estadísticos robustos aplicables en los análisis, que permiten disminuir el impacto de los *outliers*. Estos métodos logran que los resultados se vean menos afectados por la presencia de valores atípicos.

**Descartar los datos atípicos del conjunto de datos sin verificar que no se deben a un error de medición o de construcción del *dataset*, no es la solución**. Por otro lado, **sustituir estos datos por la media o la mediana, tampoco es recomendable**. Estos dos tratamientos, al igual que los tratamientos aplicados a los datos ausentes, pueden modificar los resultados obtenidos en futuros análisis, disminuir el tamaño muestral, introducir un sesgo o puede afectar tanto a la distribución como a las varianzas de la variable de interés. Si finalmente se decide eliminar o sustituir los valores atípicos, **es muy recomendable repetir los análisis con y sin valores inusuales**, para observar el efecto que ocasionan. Si el efecto es mínimo, es razonable eliminarlos o sustituirlos. Si el efecto es sustancial, no deberían ser ignorados sin justificación.

Como se ha indicado anteriormente, independientemente de la opción considerada, es importante documentar cada decisión adoptada con el objetivo de que otros analistas de datos comprendan las posibles transformaciones efectuadas sobre el conjunto de datos en cada etapa del EDA llevada a cabo.

Manteniendo el objetivo didáctico de esta guía, mostramos a continuación como se eliminan los datos atípicos por si os encontráis en la tesitura de que podéis afirmar que los valores son errores de medición o derivados de la ingesta de datos y, por tanto, susceptibles de ser eliminarlos del conjunto de datos para que no produzcan distorsiones en futuros análisis estadísticos.

*Experimenta*

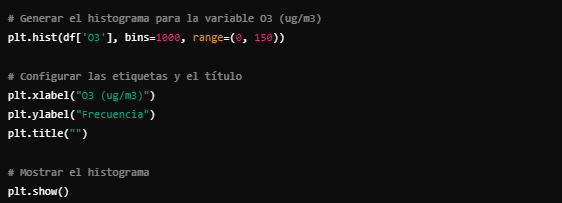
Para mostrar el proceso, debemos distinguir dos tipos de tratamiento en función del tipo de variables, continuas o discretas y categóricas.

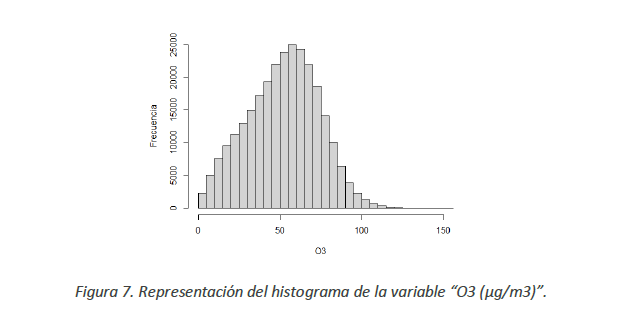
## Variables continuas

## Detección de valores atípicos

Para mostrar el proceso de detección de valores atípicos en una variable continua, utilizaremos como ejemplo la variable numérica **O3.** El proceso es exactamente igual para el resto de variables numéricas que presente la tabla.

En primer lugar, **generamos un histograma para conocer la distribución de frecuencias que presenta la variable de estudio**:

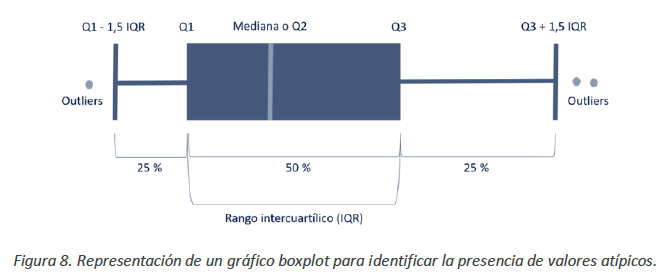




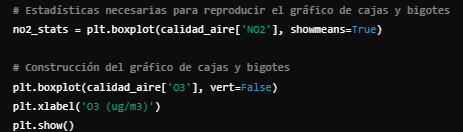
Como podemos observar en el histograma, los niveles de **O3** en el aire están mayoritariamente en un rango aproximado de, entre 0 y 100 μg/m3. Por encima de este valor la frecuencia es mínima, y podrían considerarse que es el rango de concentración de los valores atípicos. Para detectar esos valores atípicos, utilizaremos la representación más adecuada para esta tarea: un gráfico de cajas y bigotes.

**Los gráficos de cajas y bigotes o *boxplots*** (Fig. 2), **aportan una representación visual que describe la dispersión y simetría** que presentan los datos observando los cuartiles (división de la distribución en cuatro partes delimitadas por los valores 0,25; 0,50 y 0,75). Estos gráficos están compuestos por tres componentes:

* **Caja de rango intercuartílico** (*interquartile range* o IQR): Representa el 50% de los datos, comprende desde el percentil 25 de la distribución (Q1), hasta el percentil 75 (Q3). Dentro de la caja encontramos una línea que señala el percentil 50 de la distribución (Q2), la mediana. La caja aporta una idea sobre la dispersión de la distribución en función de la separación existente entre Q1 y Q3, así como también si la distribución es simétrica en torno a la mediana o si esta sesgada hacia alguno de los lados.
* **Bigotes**: Se extienden desde ambos lados de los extremos de la caja y representan los rangos del 25% de valores de la parte inferior (Q1 – 1,5 IQR) y el 25% de valores de la parte superior (Q3 + 1,5 IQR), excluyendo los valores atípicos.
* **Valores atípicos**: esta representación identifica como valores atípicos aquellas observaciones que presentan valores inferiores o superiores a los límites del gráfico (límite inferior: Q1 – 1,5 IQR y límite superior: Q3 + 1,5 IQR).



Para la obtención de los estadísticos necesarios para la representación del gráfico, recurriremos a la función **boxplots.stats()** y para la representación del gráfico utilizaremos la función **boxplot()**.



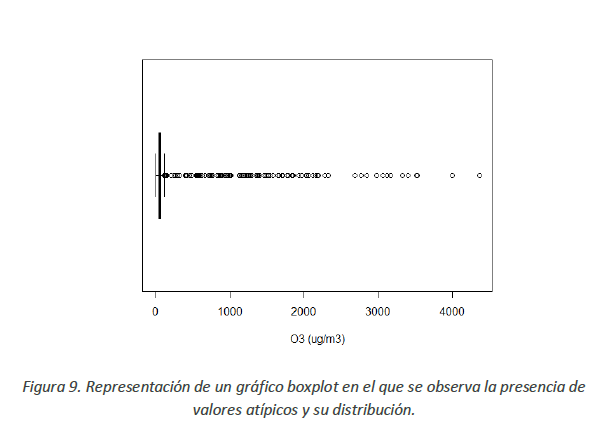
La función *boxplots.stats(),* nos muestra los estadísticos necesarios para la representación del gráfico. A continuación, mostraremos el detalle de los 4 resultados que devuelve esta función:

* + **$stats**: valores estadísticos que definen el gráfico: Q1 – 1,5IQR = 0
    - * 1. Q1 = 37
        2. Mediana o Q2 = 54
        3. Q3 = 68

Q3 + 1,5IQR = 114

* **$n**: número de observaciones que presenta la variable, en este caso 275414.
* **$out**: lista de *outliers* que presenta el *dataset* para esa variable, en este caso 468.

Usando la función *boxplot*() hemos construido el gráfico de cajas y bigotes. En este gráfico, podemos observar que la variable representada presenta una alta cantidad de datos atípicos por encima del límite superior del gráfico (Q3 + 1,5 IQR).

**

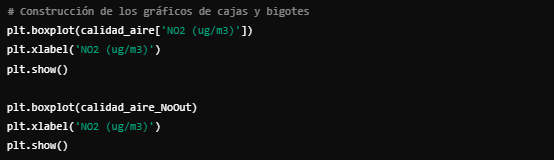
Los *outliers* detectados en esta variable coinciden con los valores más elevados de esta partícula en el aire, lo cual indica que, o bien son datos anómalos o realmente son valores extremadamente elevados de esta partícula detectados en días concretos. Un análisis exhaustivo de la calidad del aire debe contribuir a discernir esta duda, comparando también cuál ha sido el comportamiento de otras partículas en dichos días.

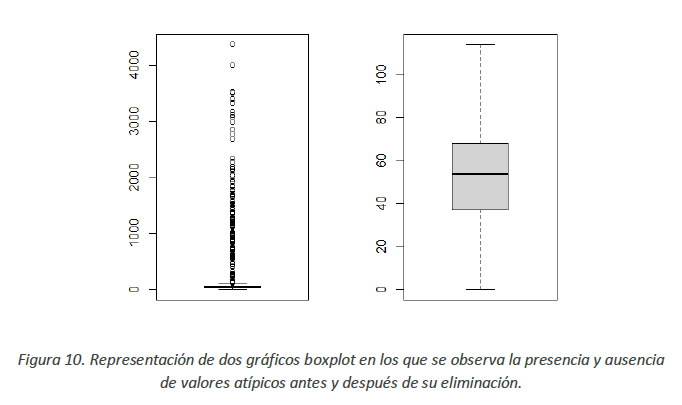
## Eliminación de valores atípicos

Una forma de eliminar los valores atípicos de una variable numérica es generar una nueva tabla, en la cual eliminemos los valores identificados como atípicos.



Una vez que hemos eliminado los *outliers*, volvemos a realizar el gráfico de cajas y bigotes con esta nueva tabla para comparar la nueva distribución de valores con la anterior. Podemos observar que tanto la mediana como los cuartiles han cambiado, así como la ausencia de valores anómalos.



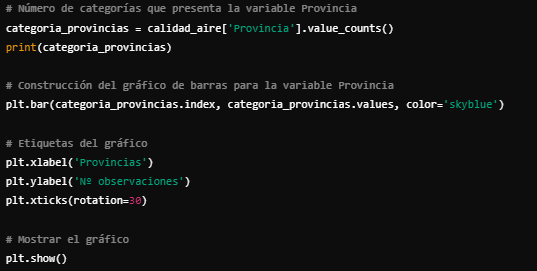


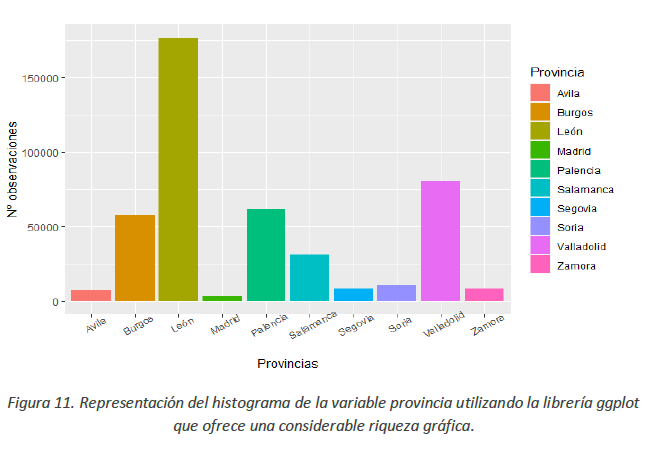
## Variables categóricas

## Detección de valores atípicos

**Al igual que en el caso anterior, para detectar si existen valores atípicos en una variable categórica, debemos conocer su distribución**. Para ello, lo más adecuado es observar su histograma representado mediante un gráfico de barras. Para ilustrar el caso, utilizaremos la variable ***Provincia***, aunque el proceso es el mismo para cualquier variable categórica.

En Python, podemos realizar este gráfico utilizando funciones básicas de la librería matplotlib, como la función **bar()**. Sin embargo, Python también ofrece otras librerías poderosas para realizar representaciones gráficas avanzadas, como seaborn y plotly, que permiten realizar visualizaciones más sofisticadas. A continuación, mostraremos cómo crear una visualización básica utilizando matplotlib, que es una de las herramientas más comunes en Python y ampliamente usada en esta guía para ilustrar ejemplos similares.



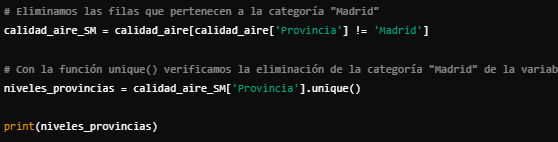


Como se puede observar en la imagen, la variable Provincia presenta 10 factores o categorías, uno de los factores puede generar dudas, es la presencia del factor llamado “**Madrid**” con muy pocas observaciones, y que evidentemente no pertenece a la Comunidad Autónoma de Castilla y León. Esta categoría podría tratarse de un error o nos podría estar indicando la presencia de una estación de calidad del aire fuera de la Comunidad Autónoma de Castilla y León, algo que se podría comprobar al disponer de las coordenadas geográficas de las estaciones en el conjunto de datos. A priori se desconoce, ya que en la descripción del conjunto de datos no se especifica y en consecuencia procede realizar esta verificación para evitar un error a la hora de realizar nuestros análisis.

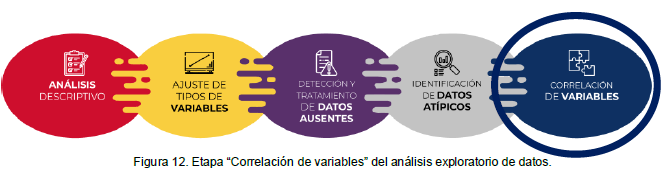
## Eliminación de valores atípicos

De acuerdo con la información de contexto derivada del análisis del conjunto de datos realizado anteriormente, se puede deducir que la categoría "*Madrid*" detectada en la variable Provincia es un valor atípico, por lo que procederemos a su eliminación.

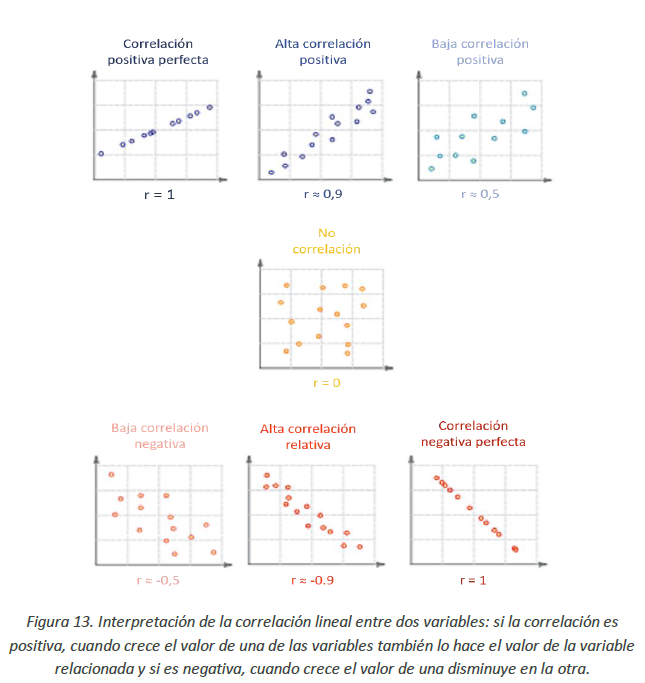
Para eliminar los valores atípicos de una variable categórica en Python, debemos eliminar la categoría que consideramos que no se ajusta a nuestros datos. En este caso, eliminaremos la categoría "Madrid" de la variable categórica Provincia utilizando las funciones de filtrado de pandas:



## Análisis de correlación entre variables

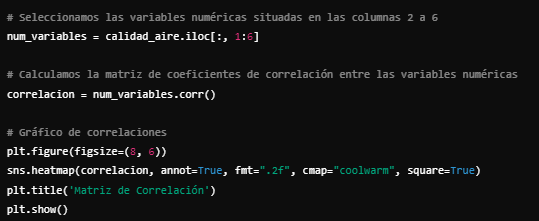


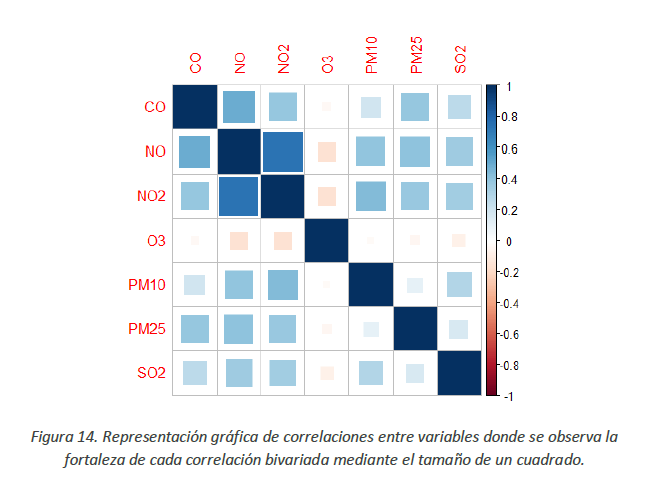
**La correlación** (valor r, en el gráfico siguiente), **determina la relación lineal entre dos o más variables**, es decir, la fuerza y la dirección de una posible relación entre variables. Dicho de otra forma, si los valores de una variable tienden a subir, los de otra u otras variables, harán lo mismo si están correlacionadas positivamente o a la inversa, si lo están negativamente. Esto no quiere decir, que una correlación entre variables indique una relación causa-efecto. De hecho, puedes encontrar cientos de ejemplos de correlaciones ficticias con las que puedes pasar un rato bien divertido. ¿En qué nos puede ayudar el análisis de correlación entre variables? La existencia de una relación fuerte en un determinado sentido entre dos variables podría inferir redundancia de información, pudiendo llegar a la eliminación de una de ellas con el fin de disminuir la complejidad en el procesamiento y análisis futuro de los datos. Esta práctica es habitual en EDA y está vinculada con la técnica de análisis de componentes principales (en muchos escritos lo verás cómo análisis PCA, por sus siglas en inglés). Sin entrar a definir esta técnica, la correlación se mide a través del **coeficiente de correlación “r” que oscila entre -1 y 1**. La correlación positiva perfecta se establece con el valor +1 e indica que los valores de las variables varían de una forma similar y la correlación negativa perfecta se establece con el valor –1, indicando que varían de forma inversa. No existe relación entre las variables, es decir, son independientes, cuando el coeficiente es 0. A continuación, se muestra una imagen que representa diferentes niveles de correlación entre variables mediante un gráfico de dispersión.



Experimenta

En primer lugar, debemos calcular la matriz de coeficientes de correlación (la fortaleza de la relación) para las variables numéricas, a partir de la cual consideramos si existe una relación entre ellas. Una vez que tengamos calculado la matriz de correlaciones, lo más habitual es mostrarlo gráficamente.





Existen múltiples formas de representar gráficamente la correlación entre variables en Python. Hemos elegido una de las más sencillas, que utiliza como recurso gráfico formas y colores para representar la fortaleza y dirección de la correlación. En el gráfico generado, observamos la correlación entre las variables analizadas. Cuando más cercano a 0 es el coeficiente de correlación rrr, los cuadrados son más pequeños y presentan un color más cercano al blanco. Observando el gráfico a simple vista, podemos hacer varias consideraciones al respecto:

* + Las variables **NO (µg/m³)** y **NO2 (µg/m³)** presentan un coeficiente alto de correlación positiva (concretamente r=0.73r = 0.73r=0.73, según la matriz de correlación). Si fuese el caso, que no lo es porque en este ejemplo queremos analizar todas las partículas que conforman la calidad del aire, este resultado nos indicaría que podríamos prescindir de una de estas variables ante la necesidad de reducir el número de variables en un análisis posterior.
  + En el resto de las variables, el coeficiente de correlación es más bajo. Tal como se observa en el gráfico, los cuadrados que representan la correlación son más pequeños y presentan un color más cercano al blanco. A simple vista, podríamos inferir que las variables **O3 (µg/m³)** y **PM10 (µg/m³)** se pueden considerar variables independientes.

Somos conscientes de que hemos hecho una introducción muy ligera del concepto de correlación, pero el objetivo de este documento es explicarte algunas tareas relevantes que se deben llevar a cabo cuando realizas un EDA. En Internet podrás encontrar mucha información sobre el tratamiento de [la correlación y la regresión lineal](https://cienciadedatos.net/documentos/24_correlacion_y_regresion_lineal) utilizando el lenguaje Python.

## Conclusiones

El Análisis Exploratorio de Datos (EDA) es un conjunto de técnicas estadísticas cuyo objetivo es explorar los datos de forma preliminar antes de aplicar cualquier proceso posterior, como una investigación científica o una visualización interactiva de datos. Este proceso es fundamental para entender los datos y las relaciones que existen entre ellos.

A través de métodos sencillos, el EDA permite realizar acciones como:

* Organizar y preparar los datos.
* Detectar fallos en el diseño y la recogida de datos.
* Tratar y evaluar los datos ausentes.
* Identificar casos atípicos.
* Analizar las posibles relaciones entre las variables.

Es crucial dedicar tiempo a aplicar estos métodos para que los resultados obtenidos de los análisis estadísticos sean altamente fiables y reflejen la realidad de los datos.

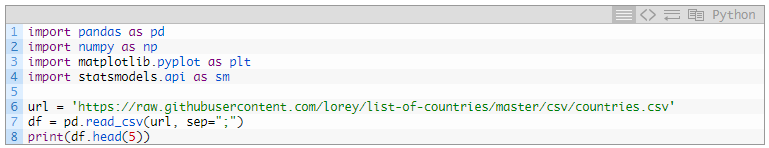
En este documento, hemos proporcionado una introducción accesible a los pasos más significativos para llevar a cabo este proceso, ilustrando su aplicación con un ejemplo utilizando datos reales del catálogo de datos abiertos de datos.gob.es. Los lectores podrán reproducir este caso práctico e incluso intentar trabajar con otros conjuntos de datos siguiendo los mismos pasos.

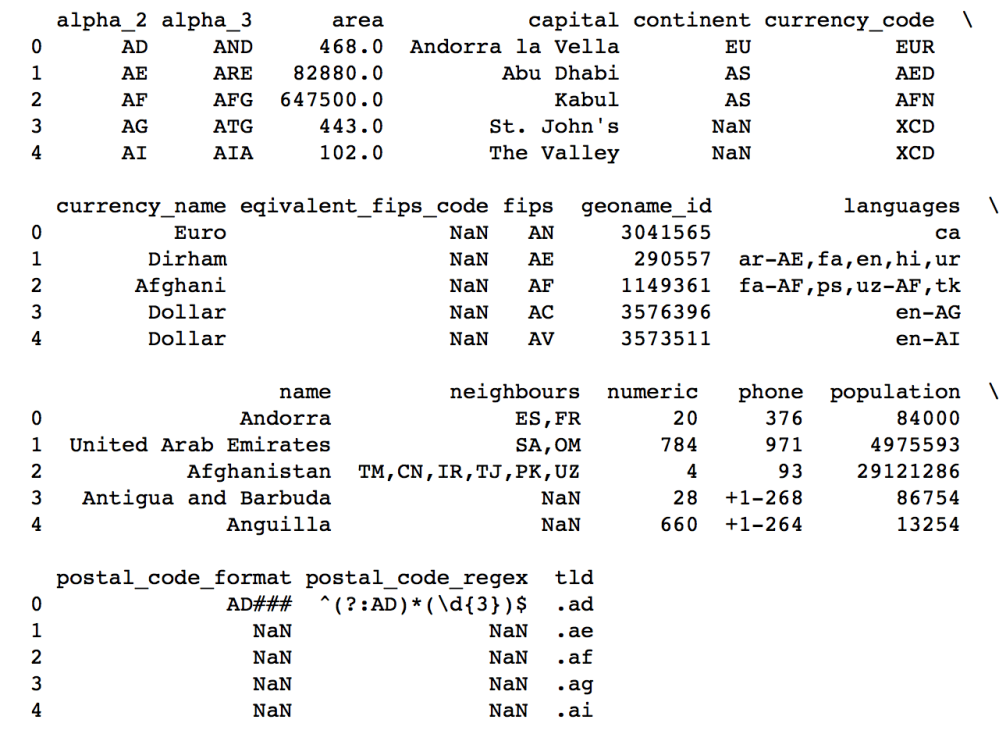
## EJEMPLO 1 - Un EDA de pocos minutos con Pandas (Python)

Vamos a hacer un [ejemplo en pandas de un EDA](https://www.aprendemachinelearning.com/analisis-exploratorio-de-datos-pandas-python/) bastante sencillo, pero con fines educativos.

Vamos a leer un csv directamente desde una URL de GitHub que contiene información geográfica básica de los países del mundo y vamos a jugar un poco con esos datos.

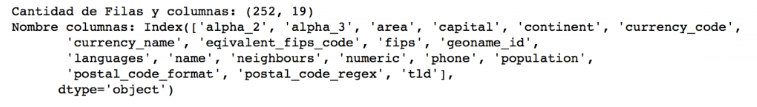
Python



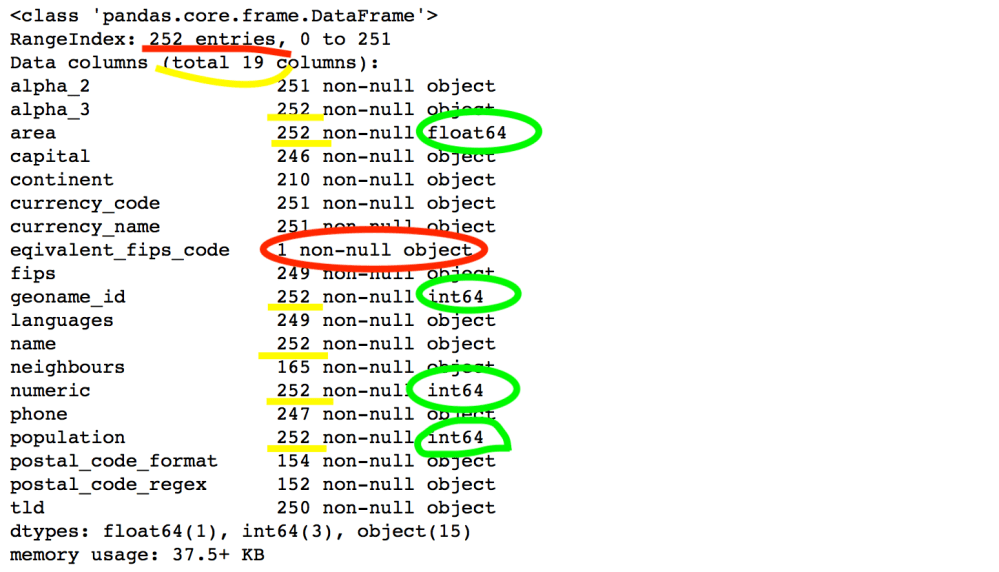


Veamos los datos básicos que nos brinda pandas:  
Nombre de columnas



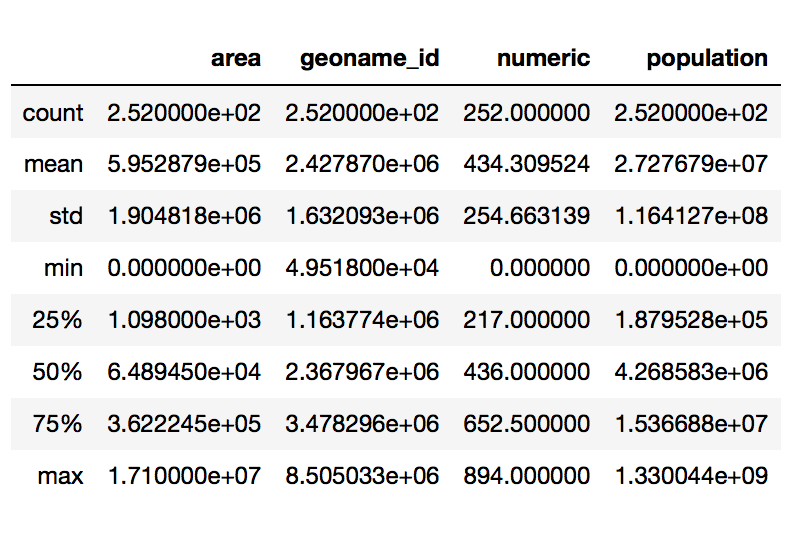
Columnas, nulos y tipo de datos

|  |  |
| --- | --- |
|  |  |

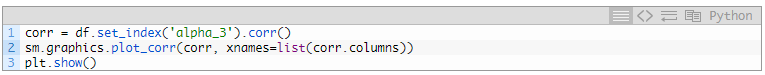


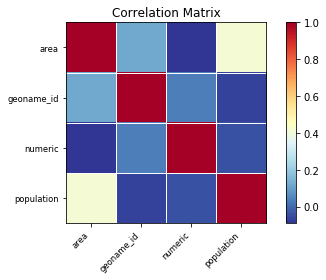
En esta salida vemos las columnas, el total de filas y la cantidad de filas sin nulos. También los tipos de datos.

descripción estadística de los datos numéricos

Pandas filtra las features numéricas y calcula datos estadísticos que pueden ser útiles: cantidad, media, desvío estándar, valores máximo y mínimo.

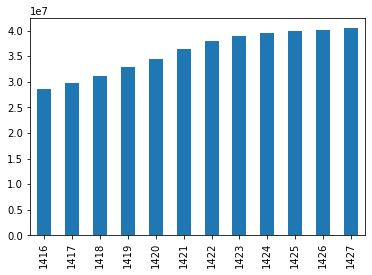
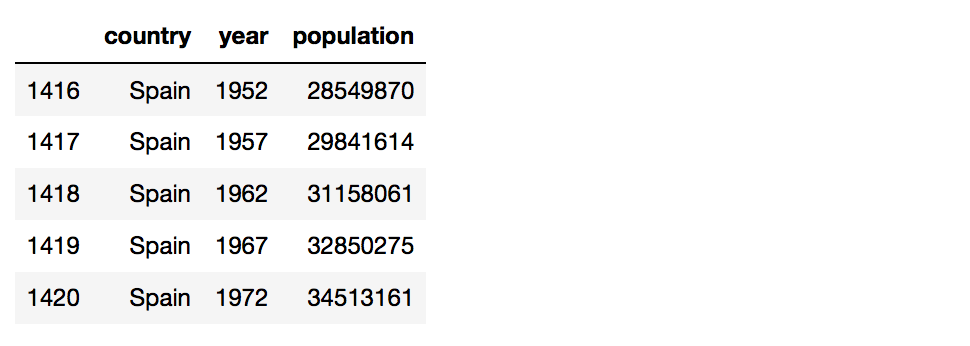
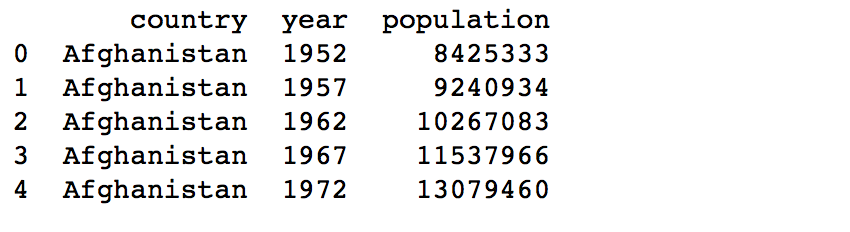
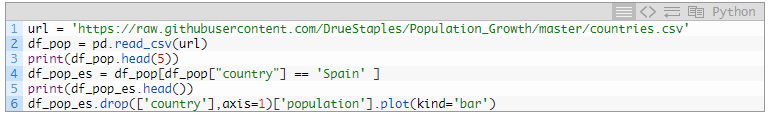
Verifiquemos si hay correlación entre los datos





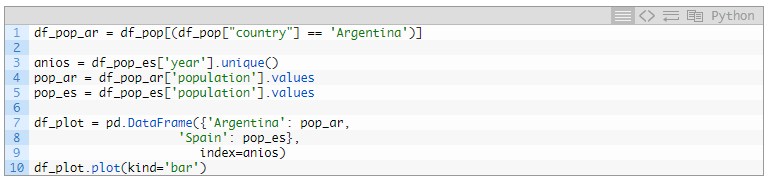
En este caso vemos baja correlación entre las variables. Dependiendo del algoritmo que utilicemos podría ser una buena decisión eliminar features que tuvieran alta correlación

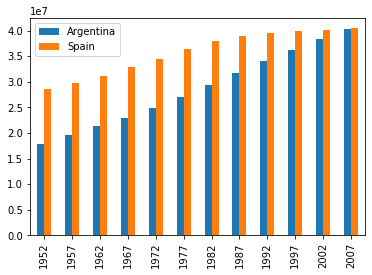
Cargamos un segundo archivo csv para ahondar en el crecimiento de la población en los últimos años, filtramos a España y visualizamos



Crecimiento de la Población de España. El eje x no está establecido y aparece un id de fila.

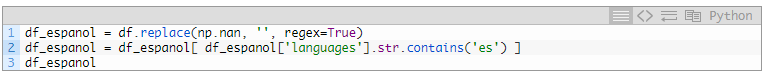
Hagamos la comparativa con otro país, por ejemplo con el crecimiento poblacional en Argentina

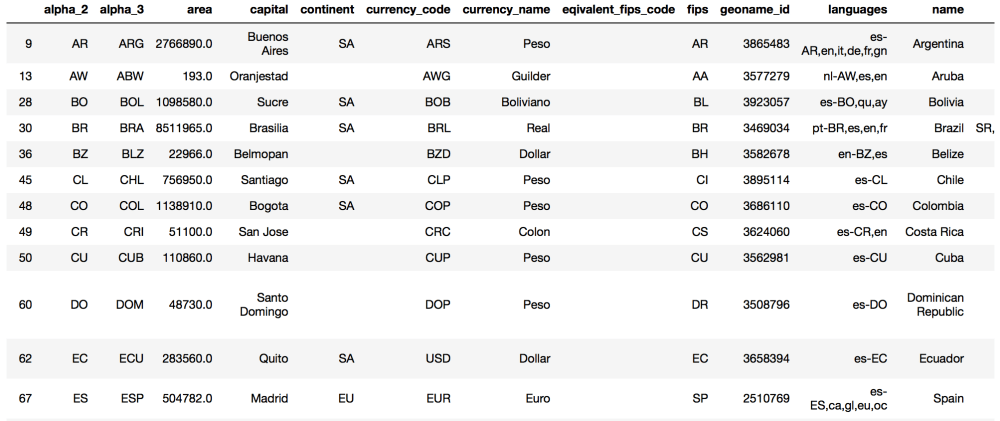




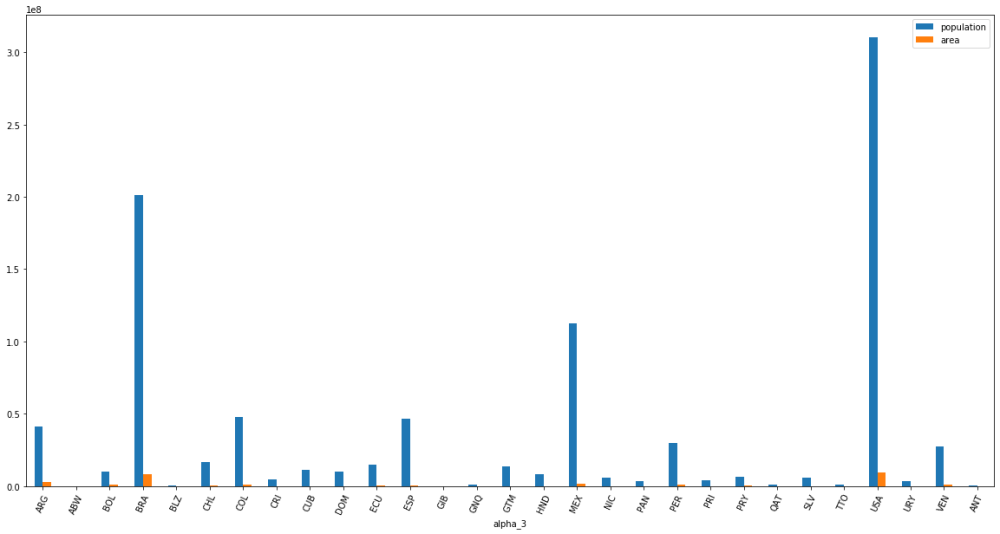
Gráfica comparativa de crecimiento poblacional entre España y Argentina entre los años 1952 al 2007

Ahora filtremos todos los paises hispano-hablantes

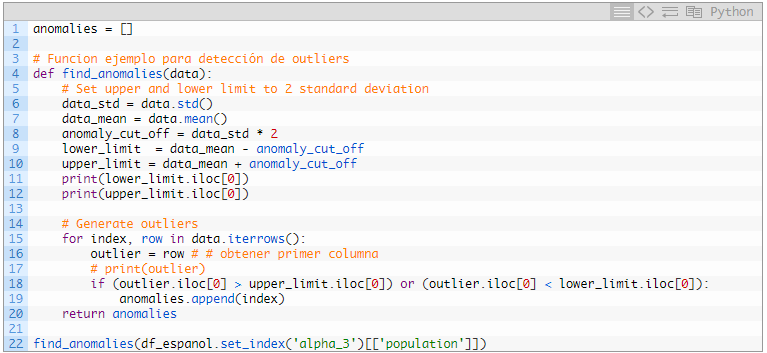


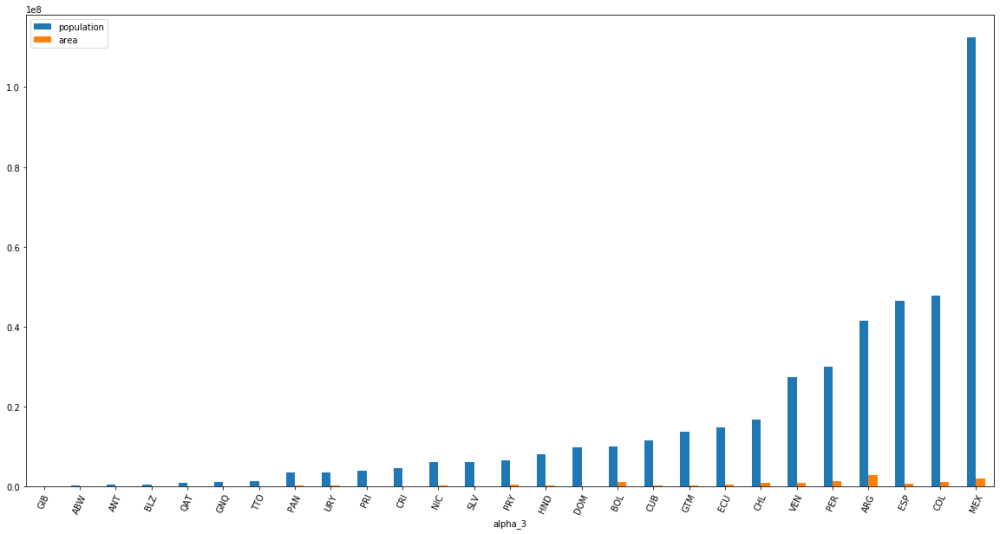
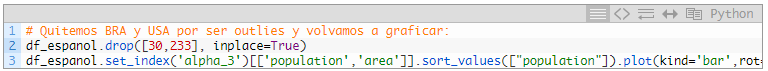


Visualizamos…



Vamos a hacer [**detección de Outliers**](https://www.aprendemachinelearning.com/deteccion-de-outliers-en-python-anomalia/)**,** (con fines educativos) en este caso definimos como limite superior (e inferior) la media más (menos) “2 veces la desviación estándar” que muchas veces es tomada como máximos de tolerancia.

Detectamos como outliers a Brasil y a USA. Los eliminamos y graficamos ordenado por población de menor a mayor.

Así queda nuestra gráfica sin outliers 🙂

En pocos minutos hemos podido responder: cuántos datos tenemos, si hay nulos, los tipos de datos (entero, float, string), la correlación, hicimos visualizaciones, comparativas, manipulación de datos, detección de ouliers y volver a graficar. ¿No está nada mal, ¿no?

## Más tareas posibles a realizar durante el análisis:

Otras pruebas y gráficas que se suelen hacer son:

* Si hay datos categóricos, agruparlos, contabilizarlos y ver su relación con las clases de salida
* Gráficas de distribución en el tiempo, por ejemplo, si tuviéramos ventas, para tener una primera impresión sobre su estacionalidad.
* Rankings del tipo “10 productos más vendidos” o “10 ítems con más referencias por usuario”.
* Calcular importancia de Features y descartar las menos útiles.