Guía práctica de introducción al Análisis Exploratorio de Datos en Python

Nov 2024

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
red.es
Iniciativa aporta
datos.gob.es
reutiliza la información pública

Este documento ha sido elaborado en el marco de la Iniciativa Aporta (datos.gob.es), desarrollada por el Ministerio para la Transformación Digital y de la Función Pública a través de la Entidad Pública Empresarial Red.es, y en colaboración con la Dirección General del Dato.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

El presente documento constituye una adaptación al lenguaje de programación Python de la guía de análisis exploratorio de datos en R publicada por la Iniciativa Aporta en el año 2021. Se mantienen el conjunto de datos y la estructura básica del análisis del análisis referenciado con la intención de facilitar la comparación entre ambos lenguajes de programación, permitir a los usuarios identificar las diferencias sintácticas y de implementación, y proporcionar un recurso valioso para aquellos que trabajan en entornos donde se utilizan ambos lenguajes. Adicionalmente, esta aproximación permite centrarse en las particularidades de cada lenguaje sin que las diferencias en los datos o en la estructura del análisis añadan complejidad innecesaria al proceso de aprendizaje. Como valor añadido, esta nueva versión incorpora secciones dedicadas a tendencias emergentes en el campo, como el análisis exploratorio automatizado y las herramientas de perfilado de datos, respondiendo así a las necesidades actuales del sector en materia de eficiencia y escalabilidad.

El análisis exploratorio de datos (AED o EDA, por sus siglas en inglés) representa un **paso crítico previo a cualquier análisis estadístico**. Su relevancia deriva tanto de la necesidad de comprender exhaustivamente los datos antes de analizarlos como de verificar el cumplimiento de los requisitos estadísticos que garantizarán la validez de los análisis posteriores.

Las técnicas estadísticas que conforman el AED permiten **desentrañar la naturaleza intrínseca de los datos, caracterizar sus atributos principales y descubrir las interrelaciones entre variables**. Este proceso sistemático sienta las bases para una comprensión profunda del conjunto de datos y fundamenta la solidez de análisis subsiguientes. La exploración inicial revela aspectos cruciales como: posibles errores en la introducción de datos, patrones de valores ausentes, y correlaciones significativas entre variables y redundancias informativas que podrían afectar a la calidad del análisis.

Paradójicamente, a pesar de su papel fundamental en la garantía de resultados consistentes y veraces, la fase exploratoria frecuentemente se minimiza en los procesos de reutilización de datos. Esta guía responde a dicha problemática presentando, tanto las **metodologías tradicionales,** como las **aproximaciones modernas al AED**, incluyendo herramientas automatizadas que facilitan la exploración sistemática de grandes conjuntos de datos.

El debate sobre la delimitación precisa de los procesos que conforman el análisis exploratorio permanece vigente en la comunidad científica. Mientras algunos expertos consideran la limpieza de datos como una fase preliminar independiente, la intrincada interrelación entre la exploración y la depuración, junto con su dependencia del contexto específico de los datos, sugiere la conveniencia de un enfoque integrado. En esta guía introductoria se detalla una serie de tareas que constituyen el conjunto mínimo a abordar para **garantizar un punto de partida aceptable para una reutilización de datos eficaz.**

Esta guía está orientada tanto a quienes se inician en el análisis de datos como a quienes buscan sistematizar los procesos del AED. Utilizando el lenguaje de programación Python[1], se ilustran los

[1] Para lograr la máxima comprensión del alcance de esta guía es recomendable tener competencias básicas en el lenguaje Python (recursos y ejercicios en Python de la Iniciativa Aporta), que es el elegido para ilustrar, mediante ejemplos, las diferentes etapas involucradas en un AED. Si no es así, animamos igualmente a continuar la lectura de esta guía dado que, como veremos a continuación, dispone de una interesante bibliografía que, además de ayudarte a entender AED, te permitirá conocer y obtener el máximo partido de este potente lenguaje de programación.

conceptos clave y se proporcionan ejemplos concretos que facilitan la comprensión y aplicación de las técnicas aprendidas.

1. METODOLOGÍA

La presente guía adopta un enfoque práctico basado en el aprendizaje experiencial, permitiendo a los usuarios **familiarizarse con técnicas de análisis mediante datos públicos reales y herramientas tecnológicas de código abierto** sin coste asociado. Todo el material desarrollado, incluyendo los datos y el código fuente, se pone a disposición pública para facilitar tanto la replicación del análisis como su adaptación a otros contextos de estudio.

Como herramienta para el **caso práctico**, se ha utilizado el lenguaje de programación Python y el entorno de desarrollo Jupyter Notebook en Google Colab. Se puede seguir la guía ejecutando las celdas del notebook publicado en el repositorio oficial, lo que garantiza la reproducibilidad de los resultados.

La selección de Python como lenguaje de programación responde a su posición destacada en el ámbito del análisis de datos, donde destaca por **combinar una sintaxis intuitiva con capacidades analíticas avanzadas**. Si bien la guía incluye los fragmentos de código necesarios para realizar cada tarea, el énfasis se sitúa en la comprensión conceptual de los procesos y en la explicación de las funcionalidades clave que Python ofrece para el análisis exploratorio. Este enfoque prioriza la claridad expositiva y didáctica sobre la optimización del código, facilitando así su comprensión por diferentes niveles de experiencia técnica.

Manteniendo la coherencia con la guía anterior, se ha escogido el mismo conjunto de datos, en concreto, **el registro de la calidad del aire en la Comunidad Autónoma de Castilla y León,** disponible en el portal datos.gob.es. Como ya se expuso en la versión implementada en R, la idoneidad de este conjunto de datos radica tanto en su relevancia social como en sus características técnicas, que lo convierten en un caso de estudio ideal para ilustrar las diferentes técnicas de análisis exploratorio.

2. ANÁLISIS EXPLORATORIO DE DATOS

Para realizar esta guía, hemos tomado como referencia el análisis exploratorio de datos descrito en el libro "Python Data Science Handbook" de Jake VanderPlas (segunda edición, 2023) disponible de forma gratuita en su primera edición, incluyendo una gran cantidad de ejemplos prácticos. El AED que proponemos seguirá los siguientes pasos:

1. **Realizar un análisis descriptivo de las variables** para obtener una idea representativa del conjunto de datos.

2. **Reajustar los tipos de las variables** para que sean consistentes al momento de realizar posteriores operaciones.

3. **Detectar y tratar los datos ausentes** para poder procesar adecuadamente las variables numéricas. Los datos ausentes son valores no registrados en algunas observaciones, y es esencial gestionarlos correctamente para evitar sesgos y problemas en el análisis.

4. **Identificar datos atípicos y su tratamiento** para evitar que puedan 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.

El gráfico (Figura 1) siguiente representa de forma esquemática el conjunto de etapas del análisis exploratorio de datos que se describe en los contenidos de esta guía.

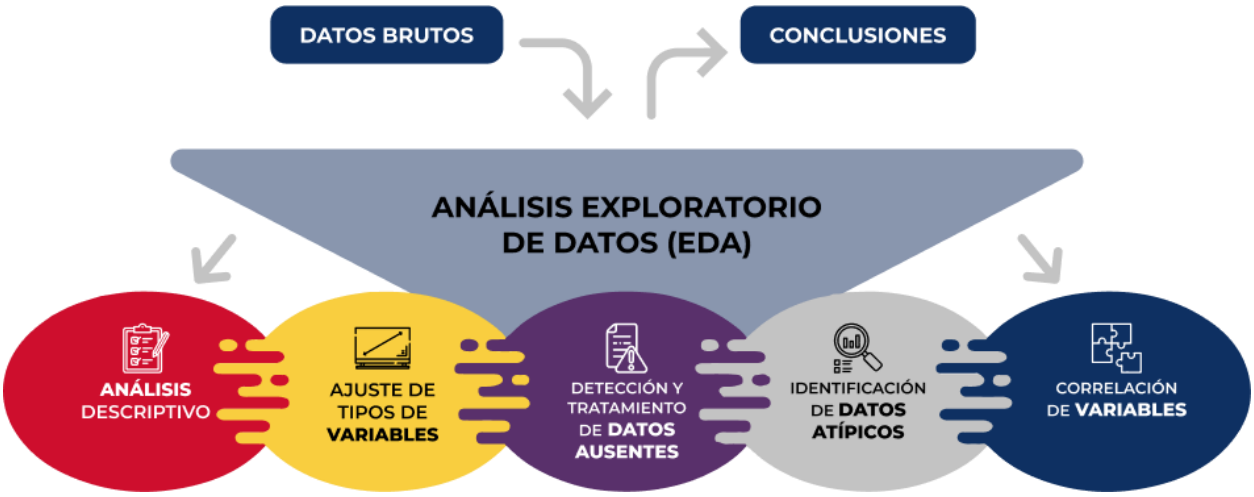

Figura 1 - Representación del conjunto de etapas del análisis exploratorio de datos

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.

2.1. ANÁLISIS DESCRIPTIVO

Figura 2 - Primera etapa del AED - 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 del catálogo de datos abiertos (puede descargarse directamente accediendo a este enlace) procederemos a realizar una **caracterización inicial del conjunto de datos** que nos permita comprender su estructura y contenido. Para ello, combinaremos dos aproximaciones complementarias: por un lado, la aplicación de técnicas de estadística descriptiva que nos proporcionarán una visión cuantitativa de las variables y sus características; por otro, la generación de visualizaciones que nos

ayudarán a comprender intuitivamente los patrones de distribución presentes en los datos. Esta fase inicial de reconocimiento sienta las bases para cualquier análisis más profundo y dirigido.

2.1.1. EXPERIMENTA

Para nuestro análisis utilizaremos el dataset mencionado anteriormente, que asignaremos al objeto **calidad_aire** en nuestro código. Este conjunto de datos servirá como base para implementar y demostrar todas las técnicas de análisis exploratorio que se presentan en la guía.

La exploración inicial se realizará mediante diversas funciones de Python especialmente diseñadas para proporcionar una **perspectiva global de la estructura y contenido de los datos**. Estas mismas funciones se utilizarán de manera recurrente a lo largo del análisis para monitorizar cómo las diferentes transformaciones y procesamientos afectan a las características del conjunto de datos.

En cuanto a la obtención inicial del conjunto de datos, se ha priorizado la automatización mediante la función read_csv(), que recupera la información directamente desde el catálogo de datos abiertos. Si bien este método puede incrementar ligeramente el tiempo de ejecución inicial, garantiza la reproducibilidad del proceso y simplifica el flujo de trabajo. Alternativamente, si se prefiere un mayor control sobre el proceso o se necesita una ejecución más ágil, existe la opción de descargar previamente el archivo desde el enlace proporcionado y cargarlo desde un directorio local.

```
# Cargar las librerías necesarias
import pandas as pd
import os

# Cargar los datos en un DataFrame
calidad_aire = pd.read_csv('https://datosabiertos.jcyl.es/web/jcyl/risp/es/medio-ambiente/calidad_aire_historico/1284212629698.csv', sep = ';')

# Mostrar las primeras filas del DataFrame
calidad_aire.head(2)
print("="*100)

# Mostrar la estructura del DataFrame
print(calidad_aire.info())
print("="*100)
# Mostrar un resumen estadístico de las variables numéricas
print(calidad_aire.describe())
print("="*100)
```

```
        Fecha  CO (mg/m3)  NO (ug/m3)  NO2 (ug/m3)  O3 (ug/m3)  PM10 (ug/m3)  PM25 (ug/m3)  SO2 (ug/m3) Provincia          Estación    Latitud  Longitud                    Posición
0  2020-12-31         0.6         8.0         16.0         NaN           6.0           NaN          1.0    Burgos           Burgos1  42.351111 -3.675556  42.3511111111,-3.67555555556
1  2020-12-31         NaN         2.0          6.0         NaN           8.0           NaN          4.0     León  C.T.L.R. - Naredo  42.816667 -5.533333  42.8166666667,-5.53333333333
================================================================================================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 446014 entries, 0 to 446013
Data columns (total 13 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Fecha         446014 non-null  object
 1   CO (mg/m3)    101158 non-null  float64
 2   NO (ug/m3)    415030 non-null  float64
 3   NO2 (ug/m3)   413497 non-null  float64
 4   O3 (ug/m3)    275414 non-null  float64
 5   PM10 (ug/m3)  344579 non-null  float64
 6   PM25 (ug/m3)  53784 non-null   float64
 7   SO2 (ug/m3)   356277 non-null  float64
 8   Provincia     446014 non-null  object
 9   Estación      446014 non-null  object
 10  Latitud       445788 non-null  float64
 11  Longitud      445788 non-null  float64
 12  Posición      445788 non-null  object
dtypes: float64(9), object(4)
memory usage: 44.2+ MB
None
================================================================================================
          CO (mg/m3)     NO (ug/m3)    NO2 (ug/m3)     O3 (ug/m3)   PM10 (ug/m3)  PM25 (ug/m3)     SO2 (ug/m3)       Latitud       Longitud
count  101158.000000  415030.000000  413497.000000  275414.000000  344579.000000  53784.000000  356277.000000  445788.000000  445788.000000
mean        0.854624      13.225808      21.409154      52.619754      22.694662     13.677172       9.092801      42.151547      -5.178965
std         0.785226      21.970729      19.108434      23.221958      17.919319     15.895495      13.790750       0.665501       1.121920
min         0.000000    -441.000000       0.000000       0.000000       0.000000      0.000000    -791.000000      38.938333      -6.781944
25%         0.300000       2.000000       8.000000      37.000000      11.000000      5.000000       2.000000      41.645556      -6.483889
50%         0.700000       5.000000      16.000000      54.000000      18.000000      9.000000       5.000000      42.542778      -4.909167
75%         1.100000      15.000000      29.000000      68.000000      29.000000     15.000000      11.000000      42.688056      -4.538333
max        25.100000     634.000000     249.000000     999.000000     557.000000    223.000000     364.000000      43.603333      -2.466667
================================================================================================
```

Figura 3 - Información general del dataset

Siendo df el nombre del objeto donde se haya guardado el DataFrame:

- `df.head()`: muestra las primeras filas del DataFrame.
- `df.info()`: proporciona una vista compacta de la estructura interna del DataFrame, indicando los tipos de variables, el número de valores no nulos y la memoria utilizada.
- `df.describe()`: muestra un resumen estadístico de las variables numéricas del DataFrame, incluyendo mínimo, máximo, media, mediana, primer y tercer cuartil, y el número de valores faltantes.

La **visualización gráfica de datos** constituye un pilar fundamental en el proceso de análisis exploratorio. Las representaciones visuales no solo complementan los análisis numéricos, sino que además revelan aspectos cruciales de los datos que podrían resultar imperceptibles mediante métodos puramente cuantitativos: patrones de comportamiento, tendencias temporales, relaciones entre variables y potenciales anomalías emergen con claridad a través de las visualizaciones apropiadas. El arsenal de herramientas visuales disponible es amplio y versátil, incluyendo histogramas para el análisis de distribuciones, gráficos de líneas para la evolución temporal, diagramas de barras para comparativas categóricas y gráficos de sectores para análisis composicionales, entre otros. La selección del tipo de visualización más adecuado para cada análisis resulta crucial y puede profundizarse en esta materia consultando esta guía sobre visualización de datos.

Particularmente, el **histograma** se destaca por su **capacidad para representar la distribución de variables numéricas.** Mediante la agrupación de datos en intervalos o "*bins*", estos gráficos revelan patrones cruciales en la estructura subyacente de los datos. La forma que adopta un histograma proporciona información esencial sobre características estadísticas fundamentales: puede mostrar si la distribución es simétrica o presenta asimetrías (sesgos positivos o negativos), si es unimodal o multimodal, si sigue aproximadamente una distribución normal o se ajusta mejor a otros modelos probabilísticos, entre otras.

La interpretación detallada de un histograma permite identificar diversos aspectos críticos para el análisis posterior:

- La tendencia central de los datos, observable a través de la ubicación del pico o picos de la distribución.
- La dispersión o variabilidad, reflejada en la amplitud y extensión de las barras.
- La presencia de colas largas o cortas, que pueden indicar la frecuencia de valores extremos.
- Posibles discontinuidades o *gaps* en la distribución, que podrían señalar problemas en la recolección de datos o fenómenos subyacentes importantes.
- La existencia de valores atípicos, visibles como barras aisladas en los extremos de la distribución.

El conocimiento de la forma de la distribución resulta crucial para la selección de métodos estadísticos posteriores, ya que muchas técnicas asumen normalidad u otras características distribucionales específicas. Por ejemplo, una distribución fuertemente sesgada podría requerir una transformación de datos o el uso de métodos estadísticos no paramétricos, mientras que una distribución bimodal podría sugerir la presencia de subpoblaciones distintas en los datos que merecen un análisis separado.

A continuación, **se generan histogramas para todas las variables numéricas** presentes en el conjunto de datos, previa importación de la librería de visualización de datos matplotlib. Mediante técnicas de programación más avanzada, como la iteración sobre las columnas del *dataset* y la automatización de la generación de *subplots*, podemos optimizar el proceso de visualización y obtener todas las distribuciones en un único lienzo. Esta aproximación programática no solo resulta más eficiente que la creación manual de gráficos individuales, sino que también facilita la detección de patrones comunes o divergencias significativas entre las distintas variables numéricas del conjunto de datos.

```
import matplotlib.pyplot as plt
import numpy as np

# Seleccionar solo las columnas numéricas
columnas_numericas = calidad_aire.select_dtypes(include=[np.number]).columns

# Calcular el número de filas y columnas para el subplot
n = len(columnas_numericas)
nrows = 3
ncols = min(n, 3)

# Crear la figura y los subplots
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(15, 5*nrows))
fig.suptitle('Distribución de Variables Numéricas', fontsize=16)

# Aplanar el array de ejes en caso de que sea 2D
axes = axes.flatten() if n > 3 else [axes]

# Crear histogramas para cada variable numérica
for i, col in enumerate(columnas_numericas):
    ax = axes[i]
    calidad_aire[col].hist(ax=ax, bins=50, edgecolor='black')
    ax.set_title(f'Distribución de {col}')
    ax.set_xlabel(col)
    ax.set_ylabel('Frecuencia')

# Ocultar subplots vacíos si los hay
```

```
for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```

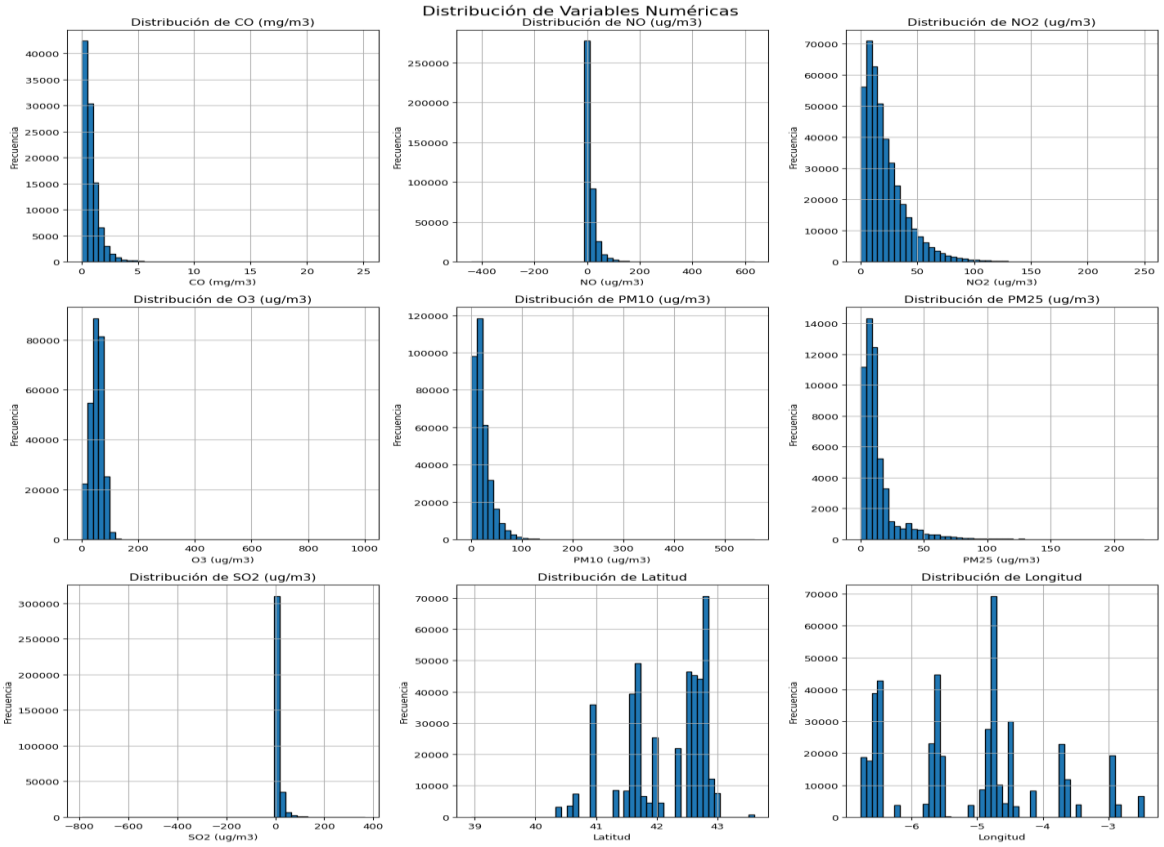

Figura 4 - Histogramas de las variables numéricas

En este conjunto de histogramas, resulta particularmente interesante analizar la distribución del NO2(µg/m³). Su histograma revela una **distribución claramente asimétrica con un sesgo positivo** pronunciado, donde la mayoría de las mediciones se concentran en valores bajos (por debajo de 50 µg/m³), pero presenta una cola larga hacia la derecha que se extiende hasta aproximadamente 250 µg/m³. Esta distribución es típica de contaminantes atmosféricos en entornos urbanos, donde se combinan períodos de concentraciones base relativamente bajas con episodios puntuales de alta contaminación, posiblemente asociados a horas pico de tráfico o condiciones meteorológicas específicas. La forma de esta distribución sugiere que podría ser necesario aplicar transformaciones logarítmicas para análisis estadísticos posteriores que asuman normalidad, y también indica la importancia de prestar especial atención a esos valores extremos que, aunque menos frecuentes, podrían representar episodios críticos de contaminación desde el punto de vista de la salud pública.

2.2. AJUSTE DE LOS TIPOS DE DATOS DE VARIABLES

Figura 5 - Segunda etapa del AED - Ajuste de tipos de variables

Tras la carga inicial resulta fundamental **verificar la correcta codificación del tipo de datos para cada variable**. Un tipo de dato inadecuado puede comprometer análisis posteriores o generar resultados erróneos. Es necesario confirmar que las variables numéricas estén efectivamente almacenadas como números (ya sean enteros o decimales), mientras que las variables cualitativas o categóricas deben estar codificadas como cadenas de caracteres y contener un conjunto finito y bien definido de categorías. Esta verificación temprana permite identificar y corregir posibles incongruencias en la tipología de los datos, como fechas almacenadas como texto o categorías codificadas erróneamente como valores numéricos.

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:** representa 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.

La correcta tipificación de las variables en un conjunto de datos no es solo una cuestión de formato, sino que resulta **fundamental para garantizar la integridad y efectividad de análisis posteriores**. La clasificación adecuada de los tipos de datos responde a principios fundamentales del análisis de datos:

- Las variables temporales (fechas) requieren un tratamiento especial que permita capturar la naturaleza secuencial y las propiedades cíclicas del tiempo.
- Las variables categóricas deben preservar la estructura jerárquica o nominal de sus categorías.
- Los datos numéricos deben mantener sus propiedades matemáticas y estadísticas.

2.2.1. EXPERIMENTA

Con la función `df.info()`, podemos realizar una primera inspección de los tipos de datos asignados a cada variable. En nuestro conjunto de datos, **encontramos distintas variables donde el tipo de dato asignado automáticamente no se corresponde con la naturaleza intrínseca de la información que contienen**, como ocurre con las variables `Fecha`, `Provincia` y `Estación`. Estas tres variables han sido codificadas genéricamente como tipo object, lo cual limita significativamente su utilidad analítica:

- La variable `Fecha` requiere una conversión a tipo fecha (*datetime*) para permitir operaciones temporales como cálculo de intervalos, agregaciones por períodos o análisis de estacionalidad. La transformación mediante `pd.to_datetime()`, no solo cambia el formato, sino que habilita todo el ecosistema de análisis temporal de Python.
- Las variables `Provincia` y `Estación` representan variables categóricas, cuya conversión a type('category') optimiza tanto el uso de memoria como el rendimiento computacional. Este tipo de datos es especialmente relevante en el análisis exploratorio ya que:
    1. Facilita la detección de categorías incorrectas o anomalías.
    2. Permite operaciones eficientes de agrupación y filtrado.
    3. Habilita análisis específicos para variables categóricas como tablas de contingencia.
    4. Optimiza la generación de visualizaciones categóricas.

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

```
# Ajustar el tipo de la variable Fecha
calidad_aire['Fecha'] = pd.to_datetime(calidad_aire['Fecha'], errors='coerce')

# Ajustar el tipo de la variable Provincia
print(calidad_aire['Provincia'].unique())
calidad_aire['Provincia'] = calidad_aire['Provincia'].astype('category')

# Ajustar el tipo de la variable Estación
print(calidad_aire['Estación'].unique())
calidad_aire['Estación'] = calidad_aire['Estación'].astype('category')
```

```
['Burgos' 'León' 'Palencia' 'Salamanca' 'Valladolid' 'Soria' 'Zamora' 'Avila' 'Segovia' 'Madrid']
['Burgos1' 'C.T.L.R. - Naredo' 'Carracedelo' 'La Robla' 'Tudela Veguin-Tudela Veguin' 'Valderas' 'Guardo' 'Hontoria 1 - Poblado'
 'Renault4' 'El Maillo' 'Salamanca5' 'Arco de ladrillo II' 'La rubia II' 'Medina del Campo' 'Michelin2' 'Puente Poniente-Mª Luisa Sánchez'
 'Vega Sicilia' 'Aranda de Duero 2' 'Miranda de Ebro2' 'C.T.L.R. - Cuadros' 'C.T.L.R. - Ventosilla' 'Lario' 'Leon1' 'Otero'
 'Toral de los Vados' 'C.T.G. - Villalba' 'Palencia 3' 'Muriel de la Fuente' 'Soria' 'Renault3' 'Zamora 2' 'Avila II' 'Burgos4'
 'Medina de Pomar' 'Leon 4' 'Ponferrada4' 'C.T.G. - Compuerto' 'Hontoria 2 - Venta de Baños' 'Salamanca6' 'Segovia 2' 'Michelin1'
 'Renault1' 'Renault2' 'VALLADOLID SUR' 'C.T.Compostilla-Congosto' 'C.T.Compostilla-Cortiguera' 'C.T.Compostilla-Villaverde'
 'C.T.Compostilla-Compostilla' 'C.T.Compostilla-Santa Marina' 'NH3' 'Miranda de Ebro1' 'Puente Regueral' 'San Martín de Valdeiglesias'
 'C.T.Anllares - Palacios del Sil' 'C.T.Anllares - Susañe' 'C.T.Anllares - Hospital del Sil' 'C.T.Anllares - Anllares'
 'C.T.Anllares - Lillo' 'C.T.Compostilla-San Miguel' 'C.T.Compostilla-Sancedo' 'C.T.Compostilla-Cueto' 'Burgos5' 'Salamanca4' 'Leon3'
 'Labradores II' 'Cementerio' 'Velilla del Rio Carrion' 'Ponferrada5' 'Ponferrada1' 'Salamanca2' 'Venta de Baños' 'Santa Teresa' 'Leon2'
 'Burgos3' 'C.T.Anllares - Anllarinos' 'C.T.Anllares - Páramo del Sil' 'C.T.Anllares - Sorbeda' 'C.T.L.R. - La Robla' 'Zamora' 'Avila'
 'Segovia' 'Aranda de Duero' 'Palencia2' 'Salamanca3' 'Burgos2' 'Ponferrada3' 'Ponferrada2' 'Miranda de Ebro3' 'Palencia1' 'Salamanca1'
 'Arco Ladrillo I']
```

Figura 6 - Categorías únicas de las variables categóricas transformadas

2.3. DETECCIÓN Y TRATAMIENTO DE DATOS AUSENTES

Figura 7 - Tercera etapa del AED - Detección y Tratamiento de Datos Ausentes

2.3.1. DETECCIÓN DE DATOS AUSENTES

La presencia de datos ausentes, también conocidos como *missing values*, `NaN` (*Not a Number*) en Python, es una situación común en muchos conjuntos de datos y **puede representar un desafío significativo en el análisis de datos**. Estos valores faltantes pueden surgir por diversas razones, como errores durante la transcripción de datos, problemas en el proceso de recolección, o incluso porque ciertos valores no estaban disponibles en el momento de la medición.

La gestión adecuada de los datos ausentes es crucial para garantizar la calidad y la fiabilidad del análisis estadístico. Los datos faltantes pueden distorsionar los resultados de los análisis, afectar la precisión de los modelos predictivos y alterar las visualizaciones gráficas, llevando a interpretaciones incorrectas o engañosas. Por ejemplo, si no se manejan adecuadamente, los datos ausentes pueden sesgar los resultados de una regresión o disminuir la capacidad predictiva de un modelo.

2.3.1.1. EXPERIMENTA

En Python, podemos usar las bibliotecas pandas y numpy para trabajar con datos ausentes. A continuación, mostramos algunas funciones útiles para detectar valores ausentes:

```
#Sección 1

# Devuelve un DataFrame booleano
calidad_aire.isna()

# Devuelve True si hay al menos un valor ausente
calidad_aire.isna().any().any()

# Devuelve el número total de NaN que presenta el DataFrame
print(calidad_aire.isna().sum().sum())

#Sección 2

# Devuelve el % de valores perdidos
print(calidad_aire.isna().mean().mean())

# Detección del número de valores perdidos en cada una de las columnas
calidad_aire.isna().sum()

# Detección del % de valores perdidos en cada una de las columnas
calidad_aire.isna().mean().round(2)
```

```
1163037
0.20058649418041727
Fecha           0.00
CO (mg/m3)      0.77
NO (ug/m3)      0.07
NO2 (ug/m3)     0.07
O3 (ug/m3)      0.38
PM10 (ug/m3)    0.23
PM25 (ug/m3)    0.88
SO2 (ug/m3)     0.20
Provincia       0.00
Estación        0.00
Latitud         0.00
Longitud        0.00
Posición        0.00
dtype: float64
```

Figura 8 - Resultados del Análisis de Valores Ausentes

Este código utiliza varias funciones de pandas:

- En la primera sección, `isna()` para generar un DataFrame booleano que indica la presencia de valores ausentes, `any()` para verificar si existe al menos un NaN en el DataFrame, y `sum()` para contar el número total de NaN.
- Además, se calcula el porcentaje de valores perdidos mediante `mean()`. Finalmente, se obtiene el número y porcentaje de valores perdidos por cada columna.

Un análisis exhaustivo de los valores ausentes en nuestro conjunto de datos revela una situación que requiere especial atención: el DataFrame "calidad_aire" **presenta 1.163.037 valores perdidos, lo que representa un significativo 20% del total de observaciones**. Este porcentaje es el mismo que el obtenido en el análisis de la guía en R, con 116.281 valores perdidos. Este hecho confirma un patrón no aleatorio en la pérdida de datos que podría estar relacionado con la disponibilidad o capacidad de los sistemas de medición.

La distribución de estos valores perdidos no es uniforme entre las variables, destacando especialmente la ausencia de datos en los parámetros de CO (mg/m³) y PM25 (µg/m³), con un 77% y 88% de valores faltantes respectivamente. La magnitud y naturaleza de estos datos faltantes plantea un desafío metodológico significativo que debe abordarse cuidadosamente para garantizar la validez y utilidad del análisis posterior. La gestión de estos valores ausentes requerirá una estrategia que equilibre la preservación de la integridad de los datos con la necesidad de mantener un conjunto de datos suficientemente completo para su análisis y reutilización efectiva.

2.3.2. TRATAMIENTO DE DATOS AUSENTES

El manejo de valores ausentes es un aspecto crucial en la preparación de datos. Existen diversas estrategias para tratar con datos faltantes, cada una con sus ventajas y desventajas. A continuación, se describen algunas de las técnicas más comunes:

- **Rellenar con estadísticos descriptivos:**
  - **Media:** sustituir los valores ausentes con la media de la variable. Es útil cuando los datos están distribuidos normalmente y no hay muchos valores faltantes.
  - **Mediana:** usar la mediana es una buena opción si los datos son asimétricos o contienen valores atípicos, ya que es menos sensible a estos extremos que la media.
  - **Moda:** reemplazar con el valor más frecuente es adecuado para variables categóricas.
- **Rellenar por valores adyacentes:**
  - **Imputación por adelante o atrás:** completar los valores faltantes con el valor de la fila o columna anterior o siguiente. Este método es útil en series temporales donde los datos suelen ser correlacionados en el tiempo.
- **Rellenar con cero:**
  - **Asignación de cero:** para valores numéricos, rellenar con cero puede ser simple, pero es generalmente desaconsejado, ya que puede introducir sesgo significativo y alterar los resultados.
- **Eliminar filas:**
  - **Eliminación de filas:** si los datos faltantes están presentes en un número pequeño de filas y el conjunto de datos es grande, se pueden eliminar estas filas. Esta técnica puede ser útil para evitar la imputación, pero se debe tener cuidado de no perder información valiosa.
- **Imputación con algoritmos de machine learning:**
  - **Modelos predictivos:** utilizar algoritmos para predecir los valores faltantes basados en otros datos. Esta técnica puede proporcionar imputaciones más precisas, pero es más compleja y computacionalmente intensiva.
- **Eliminar variables:**
  - **Variables con alta tasa de datos faltantes:** en algunos casos, puede ser apropiado eliminar variables que tengan más del 50% de datos ausentes, especialmente si la variable es poco relevante para el análisis.

La selección de la técnica de tratamiento adecuada depende del tipo de datos, la cantidad y patrón de los datos faltantes, y el contexto del análisis. Aunque la imputación por media es una técnica común, no siempre es la más adecuada. Es esencial **evaluar cómo cada método puede afectar a los resultados** y la calidad del análisis final.

Además, **es importante documentar cuidadosamente cualquier decisión tomada** en el tratamiento de datos ausentes. Un diseño riguroso del AED incluye la trazabilidad de estos procesos para poder evaluar su impacto y hacer ajustes si se detectan inconsistencias o debilidades en etapas posteriores del análisis.

2.3.2.1. EXPERIMENTA

Como ejemplo de aplicación de las opciones enumeradas, el primer tratamiento que se va a realizar sobre los datos perdidos es la **eliminación de las dos variables que presentan un porcentaje superior al 50%**, ya que un número de NaN tan alto puede producir errores o distorsionar los análisis posteriores al no ser usadas las filas que presentan NaN (en este caso, no se usaría más del 50% de las observaciones). Antes, se guarda una copia del *dataset* original, la cual será utilizada en el punto 3.

```
# Guardar copia del dataset original
calidad_aire_original = calidad_aire.copy()

# Eliminación de las variables que presentan un % de NaN superior al 50%
calidad_aire = calidad_aire.loc[:, calidad_aire.isna().mean() < 0.5]
print(f" Tras esta operación, contamos con {len(calidad_aire.columns)} columnas")
```

```
Tras esta operación, contamos con 11 columnas
```

Figura 9 - Resultados del Tratamiento de Datos Ausentes

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

```
# Seleccionamos las variables numéricas
columnas_numericas = calidad_aire.select_dtypes(include=[np.number]).columns

# Calculamos la media para cada una de las variables numéricas sin tener en cuenta los NaN
cols_mean = calidad_aire[columnas_numericas].mean()

# Sustituimos los valores NaN por la media correspondiente a cada variable
calidad_aire[columnas_numericas] = calidad_aire[columnas_numericas].fillna(cols_mean)
```

Es importante señalar que el tratamiento realizado, aunque válido como ejemplo ilustrativo, representa una **aproximación simplificada** al problema de los valores ausentes. En un análisis más exhaustivo de datos de calidad del aire, deberían considerarse aspectos adicionales como:

- **La naturaleza temporal de las mediciones:** los contaminantes atmosféricos suelen presentar patrones diarios y estacionales, por lo que la imputación por media simple podría no capturar estas variaciones cíclicas. Una aproximación más sofisticada podría considerar la imputación basada en medias móviles o valores de períodos temporales equivalentes.
- **La correlación espacial:** dado que los datos provienen de diferentes estaciones de medición, sería relevante considerar la proximidad geográfica entre estaciones para la imputación de valores ausentes, ya que estaciones cercanas tienden a registrar niveles similares de contaminación.
- **El contexto meteorológico:** la concentración de contaminantes está fuertemente influenciada por variables meteorológicas como la temperatura, precipitación, dirección y velocidad del viento. Un método de imputación más robusto podría incorporar estas variables como predictores.
- **El patrón de ausencia:** antes de eliminar variables con alto porcentaje de valores ausentes, sería conveniente analizar si estos siguen algún patrón sistemático (por ejemplo, fallos específicos de sensores o períodos de mantenimiento) que pudiera aportar información relevante sobre la calidad de los datos.

Estas consideraciones subrayan la importancia de adaptar las técnicas de tratamiento de valores ausentes al contexto específico del problema y a la naturaleza de los datos analizados.

2.4. DETECCIÓN Y TRATAMIENTO DE DATOS ATÍPICOS

Figura 10 - Cuarta etapa del AED - Identificación de Datos Atípicos

Un **valor atípico** u ***outlier*** representa una **observación que exhibe una desviación significativa respecto al patrón general de comportamiento del resto de los datos**. Estas observaciones extremas pueden surgir por diversos motivos: desde errores en la medición o registro de datos hasta fenómenos reales pero extraordinarios que merecen especial atención. Su importancia en el análisis exploratorio es crucial, ya que pueden ejercer una influencia desproporcionada en los estadísticos descriptivos, distorsionar las relaciones entre variables y comprometer la validez de modelos estadísticos posteriores. Por ejemplo, en datos de calidad del aire, un valor atípico podría representar tanto un error de medición como un episodio real de contaminación severa, lo que hace especialmente relevante su identificación y análisis contextualizado antes de tomar decisiones sobre su tratamiento. La gestión adecuada de estos valores requiere un equilibrio entre la preservación de información potencialmente valiosa y la necesidad de mantener la robustez del análisis estadístico.

El enfoque más común para el manejo de *outliers* es reducir su posible influencia en los análisis. A continuación, se mencionan algunas estrategias que se pueden considerar:

- **Métodos estadísticos robustos:** existen técnicas estadísticas robustas diseñadas para minimizar el impacto de los valores atípicos en los resultados. Estos métodos ajustan el análisis para que sea menos sensible a los *outliers*, preservando así la integridad de los resultados.

- **Eliminación de *outliers*:** eliminar valores atípicos puede ser apropiado en algunos casos, pero debe hacerse con cuidado. Antes de descartar un *outlier*, es fundamental verificar si el valor es el resultado de un error de medición o de un problema en la construcción del *dataset*.

- **Sustitución de *outliers*:** reemplazar *outliers* por la media o la mediana, por ejemplo. Aunque esta práctica puede parecer una solución sencilla, puede alterar la distribución y la varianza de los datos, introduciendo sesgo en el análisis.

Si se decide eliminar o sustituir los valores atípicos, es prudente repetir los análisis tanto con los valores originales como con los datos modificados. Esto permite observar el impacto real de los *outliers* en los resultados. Si la diferencia es mínima, puede ser razonable proceder con la eliminación o sustitución. Sin embargo, si el impacto es considerable, **se debe justificar adecuadamente cualquier decisión**.

Independientemente del enfoque adoptado, como sucede con el tratamiento de los datos ausentes, también es crucial documentar todas las decisiones tomadas durante el proceso de manejo de *outliers*.

Esto asegura que **otros analistas puedan comprender las transformaciones realizadas en el conjunto de datos** y permite una trazabilidad adecuada a lo largo del Análisis Exploratorio de Datos.

A continuación, se muestra cómo se pueden detectar y eliminar los valores atípicos, suponiendo que se puede justificar que los valores son errores de medición o problemas derivados de la ingesta de datos. El objetivo es evitar que estos valores distorsionen futuros análisis estadísticos.

2.4.1. DETECCIÓN DE VALORES ATÍPICOS

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

2.4.1.1. VARIABLES CONTINUAS

Para mostrar el proceso de detección de valores atípicos en una variable continua, utilizaremos como ejemplo la variable numérica 'O3 (µg/m³)'. El proceso es exactamente igual para el resto de las variables numéricas que presente la tabla.

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

```
plt.hist(calidad_aire['O3 (ug/m3)'], bins=100, range=(0, 150), color='blue', edgecolor='black'
)
plt.title('Distribución de O3 (ug/m3)')
plt.xlabel('O3 (ug/m3)')
plt.ylabel('Frecuencia')
plt.xlim(0,150)
plt.tight_layout()
plt.show()
```

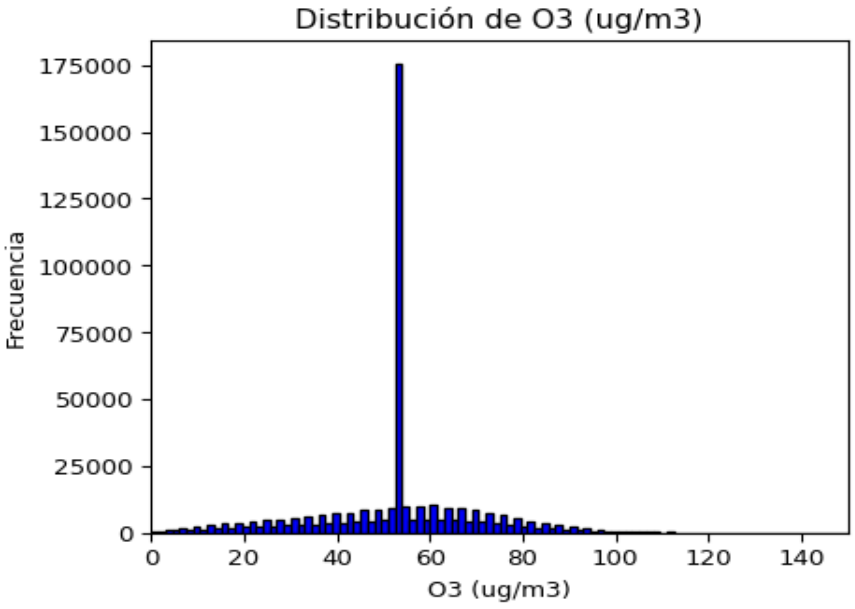

Figura 11 - Histograma de la variable 'O3 (µg/m³)'

El análisis del histograma de O3 revela una distribución asimétrica positiva característica de mediciones de contaminantes atmosféricos, donde la mayoría de las observaciones se concentran en el rango de 0 a

100 µg/m³. Las observaciones que superan este umbral presentan una frecuencia marcadamente inferior, sugiriendo la presencia de valores potencialmente atípicos. Sin embargo, en el contexto de la calidad del aire, estos valores elevados podrían representar episodios reales de alta concentración de ozono, típicamente asociados a condiciones meteorológicas específicas como alta radiación solar y temperatura elevada, especialmente durante los meses estivales. Para detectar de forma más adecuada la presencia de valores atípicos, utilizaremos la representación más adecuada para esta tarea: un gráfico de cajas y bigotes.

Los gráficos de cajas y bigotes o *boxplots* 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:

1. **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 está sesgada hacia alguno de los lados.
2. **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.
3. **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 utilizaremos la función boxplot() de seaborn, otra popular librería de visualización de datos:

```
import seaborn as sns
# Estadísticas necesarias para reproducir el gráfico de cajas y bigotes
Q1 = calidad_aire['O3 (ug/m3)'].quantile(0.25)
Q3 = calidad_aire['O3 (ug/m3)'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"Estadísticas para O3:")
print(f"Q1 - 1.5IQR = {lower_bound:.2f}")
print(f"Q1 = {Q1:.2f}")
print(f"Mediana = {calidad_aire['O3 (ug/m3)'].median():.2f}")
print(f"Q3 = {Q3:.2f}")
print(f"Q3 + 1.5IQR = {upper_bound:.2f}")
print(f"Número de observaciones: {len(calidad_aire['O3 (ug/m3)'])}")
print(f"Número de outliers: {sum((calidad_aire['O3 (ug/m3)'] < lower_bound) | (calidad_aire['O
3 (ug/m3)'] > upper_bound))}")

# Construcción del gráfico de cajas y bigotes
plt.figure(figsize=(10, 6))
sns.boxplot(x=calidad_aire['O3 (ug/m3)'])
plt.title('Gráfico de cajas y bigotes para O3 (µg/m3)')
plt.xlabel('O3 (µg/m3)')
plt.show()
```

```
Estadísticas para O3:
Q1 - 1.5IQR = 31.50
Q1 = 48.00
Mediana = 52.62
Q3 = 59.00
Q3 + 1.5IQR = 75.50
Número de observaciones: 446014
Número de outliers: 91163
```

Figura 12 - Estadísticas sobre la distribución de la variable 'O3 (ug/ m^3)'

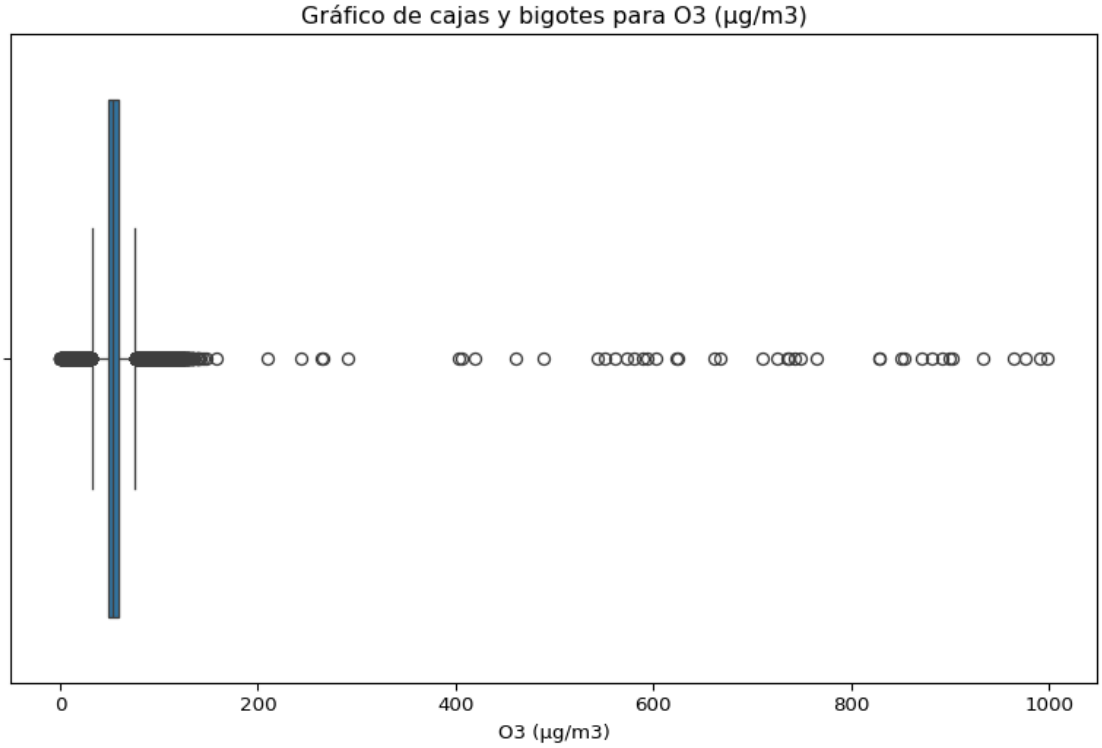

Figura 13 - Gráfico de cajas y bigotes (boxplot) de la variable ' µg/m^3'

El análisis detallado de la variable O3 revela **patrones interesantes en su distribución**. Los estadísticos descriptivos muestran una mediana de 52.62 µg/m³, con un rango intercuartílico (IQR) que va desde 31.50 µg/m³ (Q1) a 75.50 µg/m³ (Q3). El diagrama de cajas y bigotes ilustra una marcada presencia de valores atípicos por encima del límite superior, identificándose específicamente **91.163 observaciones** que superan este umbral de un total de 446.014 mediciones (aproximadamente un 20% de los datos).

Es importante señalar que este análisis representa un ejemplo ilustrativo de identificación básica de valores atípicos utilizando métodos estadísticos simples. En un estudio exhaustivo de calidad del aire, el tratamiento de estos valores requeriría consideraciones adicionales específicas del dominio y un análisis más profundo de las causas de estas desviaciones. Sin embargo, para los objetivos didácticos de esta guía, esta aproximación sirve para demostrar los conceptos básicos del análisis de valores atípicos.

2.4.1.2. VARIABLES CATEGÓRICAS

La detección de valores atípicos en variables categóricas requiere un enfoque diferente al utilizado en variables numéricas, centrándose en la **identificación de categorías inusuales o inconsistentes con el dominio del problema**. Para este análisis, la visualización mediante gráficos de barras o diagramas de frecuencia resulta especialmente útil, ya que permite identificar tanto la distribución de las categorías esperadas como la posible presencia de categorías anómalas que podrían representar errores de codificación, problemas en la recolección de datos o casos especiales que requieran atención.

En nuestro caso, utilizaremos la variable 'Provincia' como ejemplo ilustrativo de este proceso. Esta elección resulta particularmente adecuada ya que las categorías válidas están claramente definidas (las provincias de Castilla y León), lo que facilita la identificación de posibles anomalías como errores ortográficos, variaciones en el formato de escritura, o la presencia de provincias que no pertenezcan a la comunidad autónoma. Este mismo procedimiento de análisis puede y debe aplicarse a cualquier variable categórica del conjunto de datos, adaptando los criterios de validación según la naturaleza específica de cada variable.

En Python, podemos utilizar la librería matplotlib o seaborn para crear gráficos de barras y visualizar la distribución de las categorías.

```
# Número de categorías que presenta la variable Provincia
categoria_counts = calidad_aire['Provincia'].value_counts()

# Construcción del gráfico de barras para la variable Provincia
plt.figure(figsize=(10, 6))
sns.barplot(x=categoria_counts.index, y=categoria_counts.values, palette='Blues')
plt.xlabel('Provincias')
plt.ylabel('Nº observaciones')
plt.xticks(rotation=30)
plt.title('Distribución de la variable Provincia')
plt.show()
```

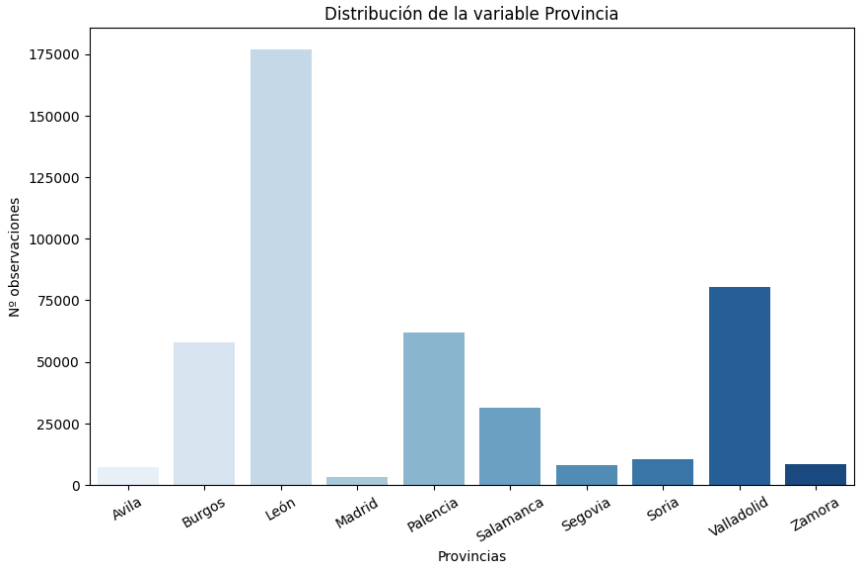

Figura 14 - Gráfico de barras sobre la variable categórica 'Provincia'

Basándonos en el análisis exploratorio, podemos deducir que la **categoría Madrid es un valor atípico** dentro de la variable 'Provincia', ya que no pertenece a la Comunidad Autónoma de Castilla y León. Procederemos en la siguiente sección a eliminar esta categoría de nuestros datos.

2.4.2. ELIMINACIÓN DE VALORES ATÍPICOS

2.4.2.1. VARIABLES CONTINUAS

Una vez identificados los valores atípicos, procedemos a eliminarlos. Una forma de eliminar los valores atípicos de una variable numérica es **generar una nueva tabla.**

```
# Se genera una nueva tabla que no contiene los valores identificados como atípicos
calidad_aire_NoOut = calidad_aire[(calidad_aire['O3 (ug/m3)'] >= lower_bound) &
(calidad_aire['O3 (ug/m3)'] <= upper_bound)]

# Construcción de los gráficos de cajas y bigotes
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

sns.boxplot(x=calidad_aire['O3 (ug/m3)'], ax=ax1)
ax1.set_title('O3 (µg/m3) con outliers')
ax1.set_xlabel('O3 (µg/m3)')

sns.boxplot(x=calidad_aire_NoOut['O3 (ug/m3)'], ax=ax2)
ax2.set_title('O3 (µg/m3) sin outliers')
ax2.set_xlabel('O3 (µg/m3)')

plt.tight_layout()
plt.show()
```

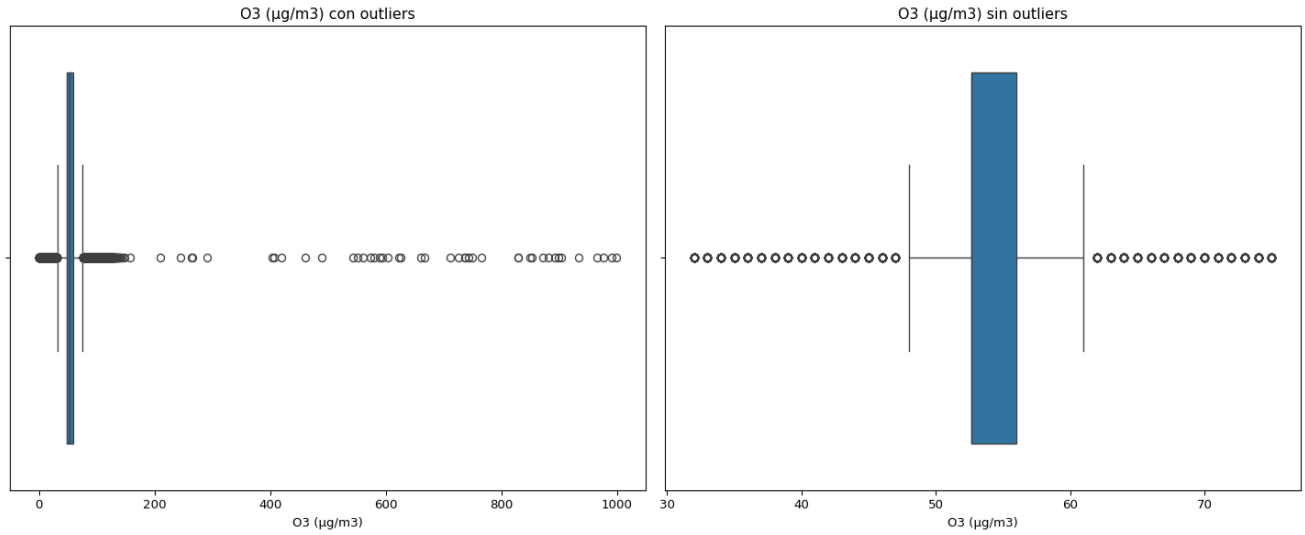

Figura 15 - Gráfico de cajas y bigotes (boxplot) de la variable 'O3 (µg/m^3)' antes y después de la eliminación de atípicos

La comparación de los diagramas de cajas y bigotes antes y después de la eliminación de valores atípicos revela cambios significativos en la distribución de la variable O3 (µg/m³). El gráfico posterior a la eliminación muestra una distribución más compacta y simétrica, con un rango considerablemente más reducido que oscila aproximadamente entre 30 y 70 µg/m³. Este cambio en la escala permite apreciar mejor la estructura central de los datos, donde tanto la mediana como los cuartiles se visualizan con mayor claridad.

2.4.2.1. VARIABLES CATEGÓRICAS

En Python, podemos utilizar el **método drop()** para eliminar filas que contienen la categoría atípica y `astype('category').cat.remove_unused_categories()` para asegurarnos de que la categoría se elimina del tipo de dato categórico.

```
# Eliminamos las filas que pertenecen al factor “Madrid”
calidad_aire_SM = calidad_aire[calidad_aire['Provincia'] != 'Madrid'].copy()

# Eliminamos el factor “Madrid”
calidad_aire_SM['Provincia'] = calidad_aire_SM['Provincia'].astype('category').cat.remove_unus
ed_categories()

# Verificamos la eliminación de la categoría "Madrid"
print(calidad_aire_SM['Provincia'].cat.categories)
```

```
Index(['Avila', 'Burgos', 'León', 'Palencia', 'Salamanca', 'Segovia', 'Soria', 'Valladolid', 'Zamora'],
dtype='object')
```

Figura 16 - Categorías únicas de la variable 'Provincia' tras la eliminación de atípicos

2.5. ANÁLISIS DE CORRELACIÓN ENTRE VARIABLES

Figura 17 - Quinta etapa del AED - Correlación de variables

La **correlación (r)** mide la relación lineal entre dos o más variables, reflejando tanto la fuerza como la dirección de su relación. En términos simples, la correlación nos indica si dos variables cambian juntas y de qué manera lo hacen:

- **Correlación positiva:** si una variable aumenta y la otra también lo hace, se dice que están correlacionadas positivamente. Un valor de (r) cercano a +1 indica una relación positiva fuerte.
- **Correlación negativa:** si una variable aumenta mientras que la otra disminuye, están correlacionadas negativamente. Un valor de (r) cercano a -1 indica una relación negativa fuerte.
- **Sin correlación:** un valor de (r) cercano a 0 sugiere que no hay una relación lineal clara entre las variables.

Es importante señalar que la correlación de Pearson, siendo la más común, no es la única medida de correlación disponible. Existen alternativas como:

- **La correlación de Spearman**, más robusta frente a valores atípicos y útil para relaciones monótonas no lineales.
- **La correlación de Kendall**, especialmente útil para muestras pequeñas y cuando los datos no siguen una distribución normal.

- **Medidas de correlación basadas en rangos**, que pueden capturar relaciones no lineales entre variables.

El análisis de correlación puede ser útil en varios aspectos:

- **Identificación de redundancia:** puede ayudar a identificar variables redundantes en un conjunto de datos. Si dos variables están altamente correlacionadas, una de ellas podría ser eliminada sin perder información significativa, lo que simplifica el análisis y el procesamiento de datos.

- **Simplicidad en el análisis:** en el contexto del AED, entender las correlaciones entre variables puede guiar la selección de variables para modelos predictivos y otros análisis estadísticos.

- **Relación con PCA:** el análisis de correlación está relacionado con técnicas como el Análisis de Componentes Principales (PCA). PCA utiliza la matriz de correlación para transformar las variables originales en un nuevo conjunto de variables, llamadas componentes principales, que capturan la mayor variabilidad en los datos.

En el contexto específico de datos ambientales y calidad del aire, el análisis de correlación cobra especial relevancia al permitir una comprensión holística de las interacciones entre contaminantes y su entorno. Esta técnica resulta fundamental para identificar **relaciones entre diferentes contaminantes** que comparten fuentes de emisión comunes, al tiempo que facilita el entendimiento de **cómo las condiciones meteorológicas influyen** en sus concentraciones. Adicionalmente, este análisis permite detectar **patrones temporales y espaciales en la distribución de contaminantes**, proporcionando una base sólida para la interpretación de la dinámica atmosférica y la gestión efectiva de la calidad del aire.

Es crucial recordar que **correlación no implica causalidad**. Aunque dos variables puedan estar correlacionadas, no necesariamente significa que una cause cambios en la otra. La correlación simplemente indica una asociación y no establece una relación causal directa.

El análisis de correlación es una **herramienta fundamental en la exploración multivariante de datos**, pero presenta limitaciones que deben ser consideradas, como coeficientes significativamente distorsionados por la presencia de valores atípicos o la **potencial omisión de relaciones no lineales y patrones complejos**. Para abordar estas limitaciones, resulta imprescindible complementar el análisis con herramientas adicionales como el uso de visualizaciones de dispersión que permitan detectar patrones no lineales, la realización de **análisis estratificado por subgrupos** para identificar posibles heterogeneidades en las relaciones o la implementación de técnicas de validación cruzada temporal o espacial que confirmen la estabilidad de las correlaciones identificadas. Este **enfoque integral y multidimensional** permite obtener una comprensión más robusta y fiable de las relaciones entre variables en un contexto profesional y completo.

2.5.1. EXPERIMENTA

En esta sección, vamos a calcular la **matriz de correlaciones para las variables numéricas** y representarla gráficamente.

```
num_variables = calidad_aire.select_dtypes(include=[np.number])

# Calculamos la matriz de coeficientes de correlación entre las variables numéricas
correlacion = num_variables.corr()

# Configuración del gráfico de correlación
```

```
plt.figure(figsize=(10, 8))

# Gráfico de correlaciones utilizando un mapa de calor
sns.heatmap(correlacion, annot=True, cmap='coolwarm', center=0, square=True, linewidths=.5, cb
ar_kws={"shrink": .5})

plt.title('Matriz de correlaciones entre variables')
plt.show()
```

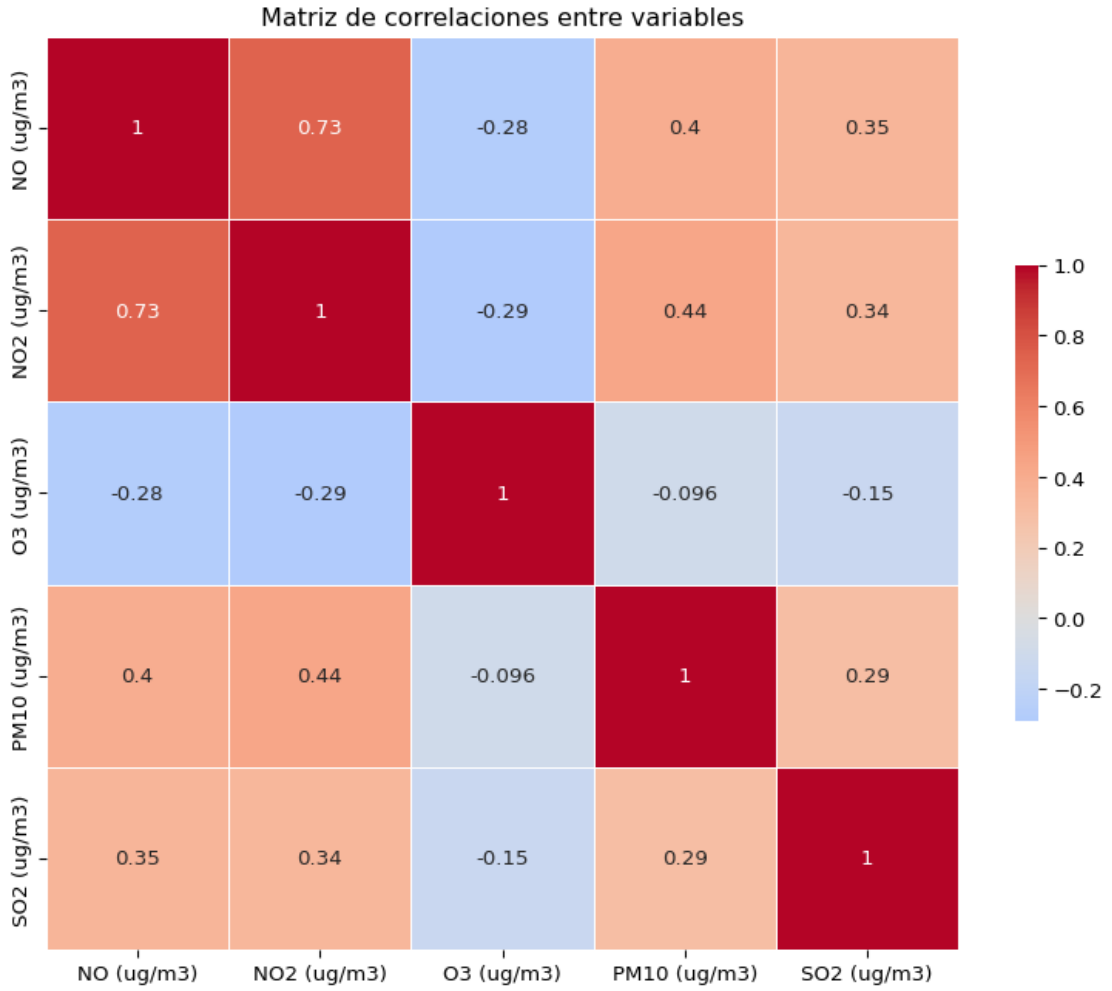

Figura 18 - Matriz de correlaciones de las variables numéricas

La visualización de correlaciones entre las variables numéricas se ha implementado mediante **un mapa de calor (*heatmap*)**, una herramienta efectiva que permite representar la matriz de correlaciones con un código de colores intuitivo, donde las tonalidades más intensas indican correlaciones más fuertes.

El análisis de la matriz de correlaciones revela varios patrones interesantes en nuestro conjunto de datos sobre calidad del aire:

Las correlaciones más significativas se observan entre NO (µg/m³) y NO_2 (µg/m³), con un coeficiente de r=0.73, sugiriendo una fuerte asociación positiva. Esta relación era esperable dado que ambos

contaminantes pertenecen a la familia de los óxidos de nitrógeno (NOx) y comparten fuentes de emisión comunes, principalmente relacionadas con procesos de combustión.

Por otro lado, **se identifican correlaciones moderadas** positivas entre PM10 (µg/m³) y varios contaminantes: con NO (r=0.4) y NO2 (r=0.44). Estas asociaciones podrían indicar una contribución común de fuentes de emisión, posiblemente relacionadas con el tráfico urbano o procesos industriales.

El resto de las variables presentan coeficientes de correlación más bajos. Esto se refleja en el gráfico, donde los cuadrados que representan estas correlaciones tienen un color más cercano al blanco. Por ejemplo, podríamos inferir que las variables O_3 (µg/m³) y PM10 (µg/m³) son relativamente independientes, ya que su coeficiente de correlación es bajo.

Estas correlaciones observadas proporcionan una base para comprender las interrelaciones entre contaminantes, aunque como se ha mencionado anteriormente, deben interpretarse con cautela y en el contexto específico de la calidad del aire urbana.

3. ANÁLISIS EXPLORATORIO DE DATOS AUTOMATIZADO

En el mundo actual, donde los conjuntos de datos son cada vez más grandes y complejos, la automatización del Análisis Exploratorio de Datos se ha convertido en una herramienta muy útil para los científicos de datos. Las bibliotecas de Python ofrecen soluciones eficientes para generar informes y visualizaciones de AED de manera automática, ahorrando tiempo y proporcionando una visión general rápida y completa de los datos.

Un AED automatizado proporciona múltiples ventajas:

- **Eficiencia y rapidez**: el AED automatizado permite procesar grandes volúmenes de datos en poco tiempo, generando informes detallados con métricas estadísticas, visualizaciones y análisis de correlación de manera automática.

- **Visión general completa**: las herramientas de AED automatizado brindan una visión panorámica de los datos, incluyendo resúmenes estadísticos, distribuciones, relaciones entre variables y posibles anomalías, facilitando la identificación de patrones y tendencias clave.

- **Detección temprana de problemas**: el AED automatizado puede ayudar a identificar problemas en los datos, como valores atípicos, datos faltantes o sesgos, en una etapa temprana del análisis, lo que permite tomar decisiones informadas sobre el preprocesamiento y la limpieza de datos.

Es fundamental, sin embargo, que esta herramienta sea utilizada teniendo en cuenta las siguientes consideraciones:

- **Interpretación de resultados**: aunque el AED automatizado proporciona una visión general rápida, es fundamental que el científico de datos interprete los resultados con criterio y conocimiento del contexto del problema.

- **Personalización**: las herramientas de AED automatizado ofrecen opciones de personalización para adaptar los informes y visualizaciones a las necesidades específicas del análisis.

- **Limitaciones**: el AED automatizado puede no ser adecuado para todos los tipos de datos o análisis. En algunos casos, puede ser necesario un análisis exploratorio más profundo y personalizado.

3.1. EXPERIMENTA

A continuación, usaremos YData Profiling, una librería que permite generar informes interactivos de AED con visualizaciones, estadísticas descriptivas, análisis de correlación y detección de valores atípicos. El informe generado con YData Profiling incluye:

- Resumen ejecutivo con información general sobre el conjunto de datos, como número de filas y columnas, tipos de datos, valores únicos, valores faltantes, etc.
- Análisis univariado de cada columna, con visualizaciones y estadísticas descriptivas como media, mediana, desviación estándar, valores mínimos y máximos, distribución, etc.
- Análisis bivariado, con matrices de correlación, diagramas de dispersión y análisis de la relación entre variables.
- Detección de valores atípicos y anomalías en los datos.
- Recomendaciones y sugerencias de mejora para el *dataset*.

En el siguiente bloque de código se muestra cómo generar un informe de AED automatizado con YData Profiling sobre el conjunto de datos calidad_aire. Este informe se genera en el directorio de trabajo del entorno de ejecución elegido y puede descargarse del repositorio de github.

```
!pip install setuptools #instalación de paquetes y dependencias
!pip install --upgrade ydata-profiling
!pip install ipywidgets

from ydata_profiling import ProfileReport
report = ProfileReport(calidad_aire_original, title='EDA automático')
report_file = 'reporte_calidad_aire.html'
report.to_file(report_file)
```

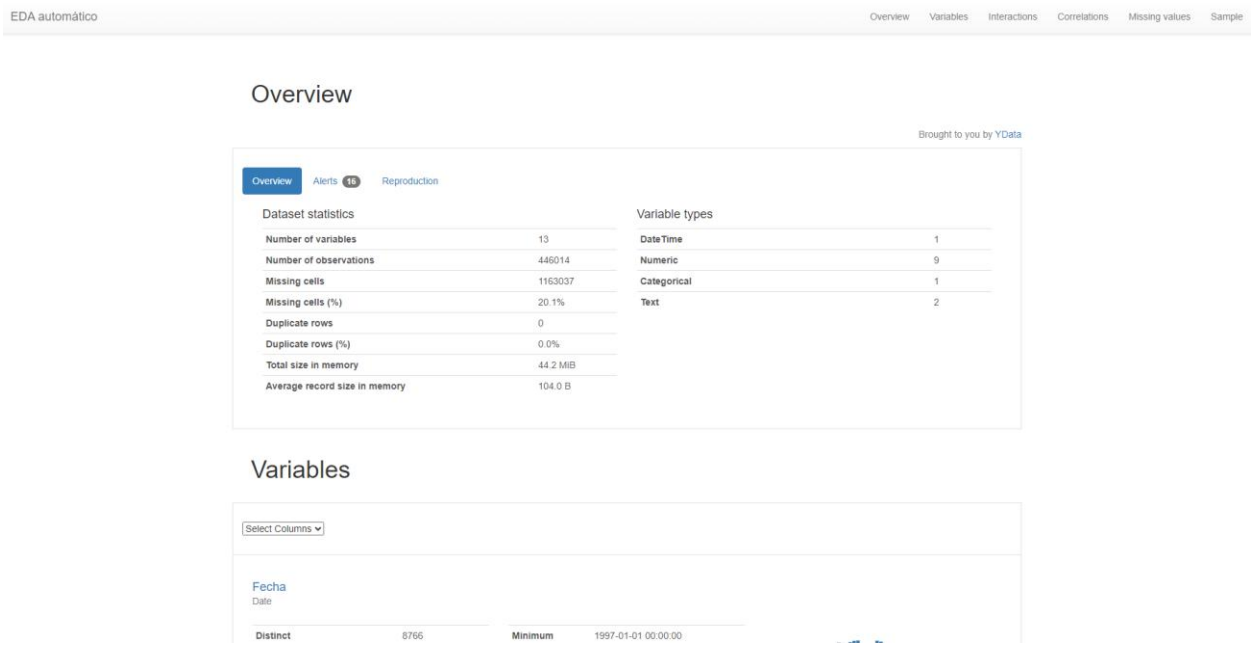

Figura 19 - Resumen general del dataset generado por 'ydata-profiling'

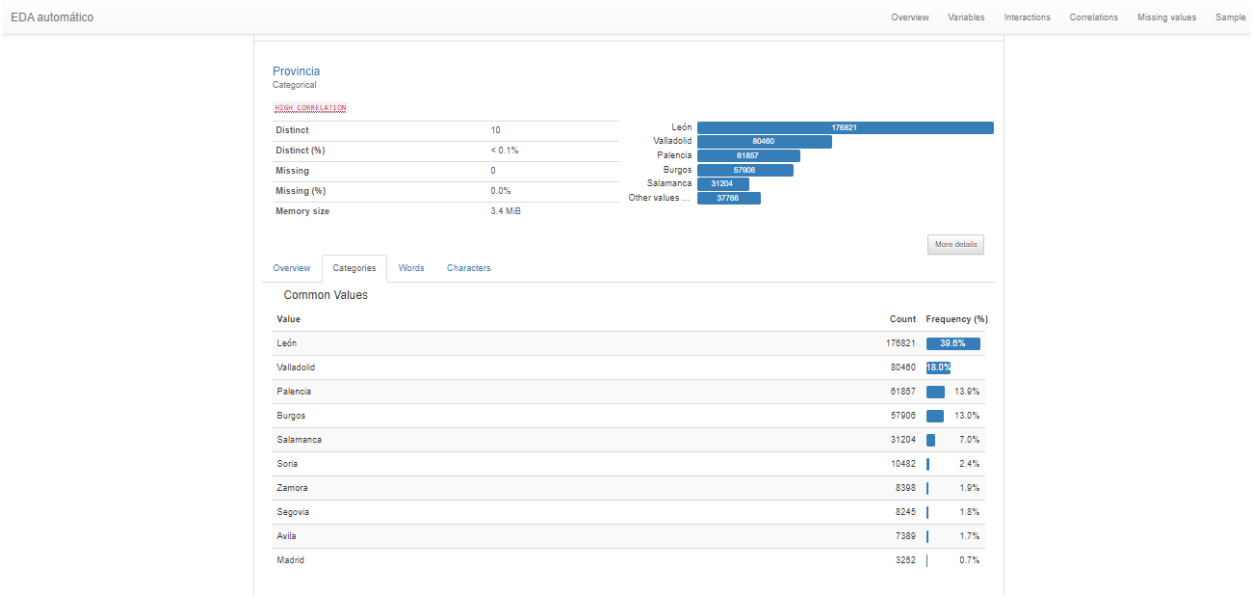

Figura 20 - Información generada automáticamente sobre la variable 'Provincia'

Podremos observar en el informe interactivo generado algunas conclusiones que se han extraído previamente, como el número de valores nulos, la alta correlación entre ciertas variables o la distribución de cada variable.

4. CONCLUSIONES

El Análisis Exploratorio de Datos (AED) constituye un pilar fundamental en la ciencia de datos moderna, proporcionando un marco metodológico robusto para la comprensión inicial de conjuntos de datos complejos. Como se ha demostrado a lo largo de esta guía, este proceso va más allá de la simple inspección preliminar de datos, constituyendo una fase crítica que determina la calidad y fiabilidad de cualquier análisis posterior, ya sea una investigación científica rigurosa o el desarrollo de visualizaciones interactivas avanzadas.

La aplicación sistemática de técnicas de AED, desde la verificación de la integridad de los datos hasta el análisis de correlaciones multivariantes, permite construir una base sólida para la toma de decisiones basada en datos.

Esta guía ha adoptado un enfoque práctico, utilizando datos reales de calidad del aire que, si bien están disponibles en el catálogo nacional datos.gob.es, se descargan directamente desde su fuente original: el portal de datos abiertos de Castilla y León. Este conjunto de datos ha permitido ilustrar tanto la aplicación de técnicas como los desafíos y consideraciones específicas que surgen al trabajar con datos ambientales. La elección de Python como herramienta de análisis responde a su creciente relevancia en el ecosistema de la ciencia de datos, proporcionando una base accesible pero potente para la implementación de estos métodos.

Es importante enfatizar que las técnicas presentadas constituyen un punto de partida básico en el análisis de datos. En aplicaciones más avanzadas, estos métodos pueden y deben complementarse con técnicas más sofisticadas, como el análisis multivariante avanzado, la detección automatizada de anomalías o el uso de técnicas de aprendizaje automático para la exploración de patrones complejos.

Esperamos que esta guía sirva como recurso práctico para aquellos que se inician en el análisis de datos, proporcionando una base metodológica sólida que pueda aplicarse y adaptarse a diversos conjuntos de datos y contextos específicos. ¡Hasta pronto!

5. PRÓXIMA PARADA

Si quieres seguir profundizando en el apasionante mundo del análisis exploratorio de los datos, te sugerimos los siguientes recursos:

- Algunos libros disponibles gratuitamente que detallan el proceso del análisis exploratorio de datos e incluyen conjuntos de datos de prueba y ejemplos con código (R o Python) para ilustrar el proceso:
  - *Python for Data Analysis*: un libro fundamental que cubre ampliamente el análisis de datos en Python, incluyendo AED.
  - *Exploratory Data Analysis with R*: un libro clásico enfocado en AED utilizando R.
  - *R for Data Science*: un recurso extenso sobre ciencia de datos y análisis exploratorio en R.
- Además de libros, la mejor forma de aprender ciencia de datos es practicando. A continuación, te dejamos enlaces a tutoriales y cursos online con una importante carga de programación práctica:
  - *Comprehensive Data Exploration with Python*: un tutorial en Kaggle que te guía a través de un completo análisis exploratorio de datos previo al entrenamiento de un modelo de aprendizaje automático.

  - *Exploratory Data Analysis with Python and Pandas*: un tutorial paso a paso sobre cómo realizar AED utilizando pandas.
  - *Exploratory Data Analysis with Seaborn*: guía completa para la visualización y AED con seaborn.
- Por último, aquí tienes algunos recursos adicionales muy útiles que, de manera gráfica, compilan la información más relevante:
  - *Data Science Cheat Sheet*: hojas de trucos de DataCamp que resumen conceptos clave de ciencia de datos.
  - *Seaborn Cheat Sheet*: ejemplos y hojas de trucos para la biblioteca Seaborn, útil en AED.
  - *Pandas Cheat Sheet*: un resumen práctico de las operaciones más comunes con Pandas en Python.