09_Groupby

November 28, 2023

0.1 Groupby

Como introducción al groupby (sí, eso ”agrupar por”) vamos a plantear unas cuestiones a nuestros datos de viajes aéreos que ya planteamos en sesiones anteriores. Recordaremos como las resolvimos y eso nos dará pie a ver una forma más eficiente de hacerlo a través de las agrupaciones con groupby y de ahí completaremos con más detalles y posibilidades adicionales.

Por tanto, como en otras sesiones, comencemos creando nuestro `DataFrame` a partir de los datos de vuelos:

[ ]:
```
import numpy as  np
import pandas as pd

df_aviones = pd.read_csv("./data/dataset_inicial_aviones.csv", index_col =␣
 ↪"Id_vuelo")
```

0.1.1 ¿Para qué groupby?

En sesiones anteriores le ”preguntamos” a nuestro `DataFrame`, `df_aviones`, por la media de consumo de los aviones por compañía, o algo parecido... y lo hacíamo de esta forma, válida, pero farragosa:

[ ]:
```
for compania in df_aviones["Aircompany"].unique():
    es_compania = df_aviones["Aircompany"] == compania
    print(f"Para la compañía {compania}:")
    for avion in df_aviones["avion"].unique():
        es_avion = df_aviones["avion"] == avion
        consumo = df_aviones.loc[es_compania & es_avion,"consumo_kg"].mean()
```

```
        print(f"Tipo <{avion}> consumo medio por vuelo <{consumo:.2f}>")
```

Esta forma aunque eficaz en términos humanos, es decir una persona tiene la info que quería, no es muy buena en términos de programacion por dos motivos: 1. El código es farragoso y aún haciéndolo una función que tuviera las columnas como parámetro sería complejo mantenerlo. 2. La salida no es muy manejable posteriormente, aunque es verdad que podríamos crearnos una estructura de salida, complicando aún más el código.

Es para este tipo de situaciones para lo que aparece el método groupby, pero ojo no solo para estas como iremos viendo en esta y las siguientes sesiones.

[ ]:

Compara los dos "trozos" de código... Sin duda es mejor el segundo ya sólo en tiempo de escritura... Pero además la salida del groupby la podemos almacenar directamente en una variable

[ ]:

Nos ha devuelto un objeto `Series` de Pandas que podemos manipular como cualquier otro `Series`:

[ ]:

[ ]:

El índice es un indice multidimensional (cada valor es una tupla) en los que no vamos a profundizar, pero que se puede operar como cualquier otro indice de una serie:

[ ]:

0.1.2 Groupby con cierto detalle

El método groupby de un `DataFrame` (también lo hay para `Series` pero eso puedes consultarlo aquí) tiene varios argumentos interesantes, el primero ya lo hemos visto la lista de columnas por las que queremos agrupar (puede ser sólo una) y otro es al argumento `as_index` que veremos luego. Antes es interesante destacar que groupby es un método "vago" [si como Jose Mota, si hay que ir se va pero ir pa ná]. Veámoslo, ejecutando solo el groupby:

[ ]:

Nos ha devuelto un objeto groupby, eso y nada de primeras es lo mismo. Para obtener algo hay que decir qué tiene que hacer con las columnas restantes:

[ ]:

En este caso, no como en el primer que previamente a la función de "agregacion" (mean en ese caso) dijimos que sólo queríamos operar sobre la columna "consumo_kg", nos devuelve un `DataFrame`. Comprobémoslo:

[ ]:

Como ya has visto podemos hacer el agrupado y luego sólo coger varias columnas y actuar sobre ellas, con diferentes funciones... Calcula ahora por Compañía y Destino el número de vuelos, venga...:

[ ]:

[ ]:

[ ]:

Pero puede que queramos ver por ejemplo la duración media del vuelo y el consumo medio realizado por Compañía, Origen, Destino:

[ ]:

Nos da error, porque si la función de agregación no se puede aplicar a alguna de las columnas que no hacen agrupación y no se han filtrado, en este caso "avion" que es un str no admite media, da un error (contar filas no da error :-)

Tendríamos que filtar previamente como hemos hecho con "consumo_kg", y quedarnos con las columnas que queremos o por lo menos con aquellas para las que la función final que aplicamos (mean, en este caso) sea válida

[ ]:

[ ]:

Pero son muchos resultados, ¿como filtro? Pues aunque no hablermos mucho más de multindice, miremos varios ejemplos:

[ ]:
```
# Resultados para la compañía Airnar


# Resultados para todos los vuelos París-Cádiz


# Resultados para los vuelos a Ginebra y Nueva York de FlyQ
```

Nosotros, en general, si queremos conservar la salida de un groupby para seguir procesándola, emplearemos el argumento `as_index` con valor "False"

[ ]:
```
resultado_no_index = df_aviones.groupby(["Aircompany","Origen","Destino"],␣
↪as_index = False)[["consumo_kg","duracion"]].mean()
```

[ ]:

Porque de esta forma seguiremos teniendo toda la información en columnas, eso sí como se han agrupado observa que los índices originales (los identificadores del vuelo) pierden sentido y se pierden...