05_Duplicados

November 28, 2023

0.1 Duplicados

ANtes de empezar con manipulacion de datos con funciones definidas por el usario, vamos atrabajar con los duplicados. Los duplicados en los datos (generalmente de toda una fila), no son necesariamente malos, puede que los datos necesiten precisamente tener esos duplicados. Pero en general siempre va a ser bueno saber si existen, y en el caso de no quererlos saber como eliminarlos.

Como ultimante, nos cargamos un `DataFrame` de referencia, que esta vez es algo especial porque sí contiene duplicados, aunque lo usaremos poquito:

[1]:
```
import numpy as  np
import pandas as pd

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

0.1.1 Detección y eliminación de filas duplicadas

La forma de detectar filas duplicadas es emplear el método duplicated. El método se puede aplicar a todo el `DataFrame` o a una selección en la que nos dirá si hay duplicados en esa selección y tb se puede aplicar a series.

La sintaxis para comprobar por filas es:8 comprbar si hay una fila completemante duplicada

```
df_aviones.duplicated() # con un parámetro importante keep, que por defecto está a "first" ( K
```

Este método devuelve una serie de booleanos donde True dice que la fila se considera duplicada y False que no se considera duplicada.

El criterio de Pandas para etiquetar una fila como duplicado depende del mencionado argumento `keep`: 1. Etiqueta todas las filas que están duplicadas como duplicadas. (keep = False) 2. Etiqueta como duplicadas todas las apariciones menos la primera. (keep = "first") 3. Etiqueta como duplicadas todas las apariciones menos la última (keep = "last") .

¿Y para qué tanto lío? Porque cuando queramos eliminar los duplicados, se eliminarán solo los etiquetados como tal y así de esta forma me puedo quedar con las primeras apariciones, o las últimas. Eligiremos first o last depende del orden que queramos en nuestro set de Datos

Por ejemplo, supongamos un dataframe como:

[4]:
```
df_dup = pd.DataFrame( {"col1": ["a","c","a","a","h","c"],# al usar un␣
 ↪diccionnario usa las claves para nombrar las columnas
                         "col2":["b","d","b","b","j","d"],
                         "col3":["c","a","c","c","k","a"]},
                         index =␣
 ↪["fila1","fila2","fila3","fila4","fila5","fila6"])# pa nombra las filas
df_dup
```

[4]:
```
       col1 col2 col3
fila1    a    b    c
fila2    c    d    a
fila3    a    b    c
fila4    a    b    c
fila5    h    j    k
fila6    c    d    a
```

Donde las filas 1,3 y 4 están duplicadas por un lado, y las filas 2 y 6 por otro. Veamos el efecto de duplicated:

[5]:
```
df_dup.duplicated() # es igual a df_dup.duplicated(keep = "first")
# aqui nos dice que los duplicados son la 3,4,y la 6 , dejando la 1,2 y 6 como␣
 ↪buenas
```

[5]:
```
fila1    False
fila2    False
fila3     True
fila4     True
fila5    False
fila6     True
dtype: bool
```

Y si aplicamos el método como condición:(como serie loc)

[7]:
```
df_dup.loc[df_dup.duplicated()]# aqui nos marca la 3,4 6 que son repetidas y no␣
 ↪muestra mas
```

[7]:
```
       col1 col2 col3
fila3    a    b    c
fila4    a    b    c
```

```
fila6    c    d    a
```

Con el **Keep** lo que queremos decir es quedate es decir le decimos con que valores nos vamos a quedar no los que eliminara

Ahora con keep = "last"

[9]:
```
df_dup.duplicated(keep = "last")# ahora ha puesto las primeras como repetidas y␣
↪la 4,5, 6 como buenas
```

[9]:
```
fila1     True
fila2     True
fila3     True
fila4    False
fila5    False
fila6    False
dtype: bool
```

Y de nuevo veamos que nos selecciona como duplicados:(como metodo con loc como condicion)

[12]:
```
df_dup.loc[df_dup.duplicated(keep = "last")]# considera como duplicados la␣
↪1,2,3 y nos quedamos con los ultimos(last) como buenos
```

[12]:
```
      col1 col2 col3
fila1    a    b    c
fila2    c    d    a
fila3    a    b    c
```

Y finalmente con keep = False:

[15]:
```
df_dup.duplicated(keep = False)# considera las duplicadas a true y a false la␣
↪buena
```

[15]:
```
fila1     True
fila2     True
fila3     True
fila4     True
fila5    False
fila6     True
dtype: bool
```

Y el `DataFrame` seleccionado según duplicado:(u si lo vemos como condicion en loc)

[16]:
```
df_dup.loc[df_dup.duplicated(keep = False)]# Todo menos la fila 5 nos da como␣
↪duplicado
```

[16]:
```
      col1 col2 col3
fila1    a    b    c
fila2    c    d    a
fila3    a    b    c
```

```
fila4    a    b    c
fila6    c    d    a
```

Y todo esto es importante, porque el método para eliminar duplicados sólo lo hara de aquellas que marquemos como duplicados y también tiene su parámetro `keep` que implica la misma filosofía

Veámoslo con df_dup y luego apliquemos a nuestro `DataFrame` de aviones

(Primero nos hacemos una copia de backup)

[17]:
```
df_dup_reserva = df_dup.copy()
```

[18]:
```
df_dup.drop_duplicates() #keep = "first"# nos quedamos con las primera␣
↪aparaciones
```

[18]:
```
      col1 col2 col3
fila1    a    b    c
fila2    c    d    a
fila5    h    j    k
```

[19]:
```
df_dup.drop_duplicates(keep = "last") # se queda con las ultimas y borra las␣
↪primeras
```

[19]:
```
      col1 col2 col3
fila4    a    b    c
fila5    h    j    k
fila6    c    d    a
```

[20]:
```
df_dup.drop_duplicates(keep= False)# borrra todos los que estan duplicados y se␣
↪queda con el unico que no esta duplicado
```

[20]:
```
      col1 col2 col3
fila5    h    j    k
```

[21]:
```
df_dup# drop no modifica el df original para ello tendremos que usar el inplace␣
↪= True
```

[21]:
```
      col1 col2 col3
fila1    a    b    c
fila2    c    d    a
fila3    a    b    c
fila4    a    b    c
fila5    h    j    k
fila6    c    d    a
```

[ ]:
```

```

Por cierto, si queremos que el método modifique el `DataFrame` que lo llama debemos usar el argumento `inplace` con valor a True

[25]:
```
df_dup.drop_duplicates(keep = False, inplace = True)# borra todos los␣
↪dupplicados y borralos del original  tb
```

[26]:
```
df_dup# nos ha dejado solo la unica fiola no dupliacda
```

[26]:
```
      col1 col2 col3
fila5    h    j    k
```

Si lo aplicamos a nuestro dataframe

[28]:
```
# Veamos algunos duplicados
df_aviones.loc[df_aviones.duplicated(keep = False)] # le marcamos False para␣
↪que nos diga todos los duplicados
# como son mucghas filas vamos a filtrar un indice
df_aviones.loc["Mol_PaBa_10747"]
```

[28]:
```
                Aircompany Origen Destino  Distancia       avion  consumo_kg  \
Id_vuelo
Mol_PaBa_10747  MoldaviAir  París    Bali      11980  Boeing 747  130260.936
Mol_PaBa_10747  MoldaviAir  París    Bali      11980  Boeing 747  134092.140
Mol_PaBa_10747  MoldaviAir  París    Bali      11980  Boeing 747  140477.480
Mol_PaBa_10747  MoldaviAir  París    Bali      11980  Boeing 747  135369.208
Mol_PaBa_10747  MoldaviAir  París    Bali      11980  Boeing 747  130260.936
Mol_PaBa_10747  MoldaviAir  París    Bali      11980  Boeing 747  134092.140

                duracion
Id_vuelo
Mol_PaBa_10747       818
Mol_PaBa_10747       818
Mol_PaBa_10747       818
Mol_PaBa_10747       818
Mol_PaBa_10747       818
Mol_PaBa_10747       818
```

[30]:
```
# Podemos ver los duplicados, filtrando por otros campos, veamos los de FlyQ␣
↪usando para ellos las series loc para establecer una condicion
df_aviones.loc[(df_aviones.Aircompany =="FlyQ") & (df_aviones.duplicated(keep=␣
↪False))] # aqui keep quere decir "no te queds con ningun duplicado"
df_aviones.loc["Fly_BaRo_10747"] #filtramos por indice
```

[30]:
```
                Aircompany     Origen Destino  Distancia       avion  \
Id_vuelo
Fly_BaRo_10747       FlyQ       Bali    Roma      12738  Boeing 747
Fly_BaRo_10747       FlyQ       Bali    Roma      12738  Boeing 747
Fly_BaRo_10747       FlyQ       Bali    Roma      12738  Boeing 747
Fly_BaRo_10747       FlyQ  Barcelona    Roma        859  Boeing 747
Fly_BaRo_10747       FlyQ  Barcelona    Roma        859  Boeing 747
Fly_BaRo_10747       FlyQ       Bali    Roma      12738  Boeing 747
```

```
Fly_BaRo_10747       FlyQ  Barcelona    Roma        859  Boeing 747
Fly_BaRo_10747       FlyQ       Bali    Roma      12738  Boeing 747
Fly_BaRo_10747       FlyQ  Barcelona    Roma        859  Boeing 747
Fly_BaRo_10747       FlyQ  Barcelona    Roma        859  Boeing 747
```

Id_vuelo	consumo_kg	duracion
Fly_BaRo_10747	141218.563200	1049
Fly_BaRo_10747	141218.563200	1049
Fly_BaRo_10747	155340.419520	1049
Fly_BaRo_10747	9431.648200	77
Fly_BaRo_10747	9248.509400	77
Fly_BaRo_10747	144042.934464	1049
Fly_BaRo_10747	9706.356400	77
Fly_BaRo_10747	149691.676992	1049
Fly_BaRo_10747	9248.509400	77
Fly_BaRo_10747	9797.925800	77

[31]:
```
# Finalmente veamos el agregado de duplicados en función del parámetro keep
for val_keep in [False, "first", "last"]:# itera con la variable val_keep para␣
 ↪ver [ lo contenido en la lista]
    num_vuelos= len(df_aviones.loc[df_aviones.duplicated(keep = val_keep)])#␣
 ↪dime el numero de vuelos que hay duplicados en el DF con valores que␣
 ↪iteramos en el For
    print(f"Para Keep ={val_keep}")# aqui imprimira los valores asignados en el␣
 ↪for a keep
    print(f"Nuemro de registros:{num_vuelos}")# imprimira el numero de vuelos ␣
 ↪dduplicados con las 3 condiciones citadas en for
```

```
Para Keep =False
Nuemro de registros:411
Para Keep =first
Nuemro de registros:215
Para Keep =last
Nuemro de registros:215
```

Nos dice que tenemos esos duplicados considerandolos todos. Ahora nosotros tenemos que decidir qué hacemos... [pero eso para la sesión en vivo]

0.1.2 Detección de columnas

Para terminar veamos como se pueden detectar duplicados en una o varias columnas, siempre teniendo en cuenta que este caso es aún más frecuente que el anterior y, en general, salvo para columnas que deban tener valores únicos (como por ejemplo el DNI de una persona), no será muy útil comprobarlo.

[32]:
```
df_dup = df_dup_reserva.copy() # recuperamos el anterior
df_dup
```

[32]:
```
      col1 col2 col3
fila1    a    b    c
fila2    c    d    a
fila3    a    b    c
fila4    a    b    c
fila5    h    j    k
fila6    c    d    a
```

[33]:
```
df_dup["col2"].duplicated()# exoime los primeros
```

[33]:
```
fila1    False
fila2    False
fila3     True
fila4     True
fila5    False
fila6     True
Name: col2, dtype: bool
```

[34]:
```
df_dup ["col2"].duplicated(keep = False)# no se queda bno ninguno duplicado
```

[34]:
```
fila1     True
fila2     True
fila3     True
fila4     True
fila5    False
fila6     True
Name: col2, dtype: bool
```

[35]:
```
# tb podria ver el DF filtrado por esa condicion con una serie panda
df_dup.loc[df_dup["col2"].duplicated(keep = False)] # no te quedes sin nunguna␣
 ↪dupl( las que muestra son las dupliocadas si comparamos por loc)
# toda funcion , todo metodo, toda comprobacion que devuelva una serie alineada␣
 ↪con un DF, es decir con el nombre de su filas y un booleano , me va a servir␣
 ↪para filtra ese DF
```

[35]:
```
      col1 col2 col3
fila1    a    b    c
fila2    c    d    a
fila3    a    b    c
fila4    a    b    c
fila6    c    d    a
```

Y si queremos ver varias columnas

[36]:
```
# recuerda para coger varias columnas , el fantasy indexado, abre dobles␣
 ↪corchetes y pon la liista con las col que quierrtas y en el orden que quieras
```

```
df_dup[["col2","col1"]].duplicated(keep ="last") #esto nos hace un␣
 ↪subdataframe, y nos dira las filas que tiene las col.1 y 2 repetidas con el␣
 ↪keep = "last"
#las true son las repetiodas por el last
```

[36]:
```
fila1     True
fila2     True
fila3     True
fila4    False
fila5    False
fila6    False
dtype: bool
```

Si ahora quisieramos eliminar esas filas, usariamos drop_duplicates también pero ojo nos cargaríamos toda la fila y a lo mejor no es lo que nos interesa.

[38]:
```
##vamos a borrar esas filas por esa condicion con una serie panda (condciion =␣
 ↪[df_dup[["col1", "col2"]].duplicated(keep= "last")])
df_dup.loc[df_dup[["col1", "col2"]].duplicated(keep= "last")].
 ↪drop_duplicates(keep= "last")
```

[38]:

	col1	col2	col3
fila2	c	d	a
fila3	a	b	c

[ ]:
```

```