1. Filtrado y selección de datos

January 8, 2023

1 Transformación de datos

1.0.1 Filtrado y selección de datos

Carga de librerías

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

Importado de datos

[2]:
```
df = pd.read_csv("nycflights.csv")
#df.info()
df.head()
```

[2]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	\
0	2013	6	30	940	15	1216	-4	VX	N626VA	
1	2013	5	7	1657	-3	2104	10	DL	N3760C	
2	2013	12	8	859	-1	1238	11	DL	N712TW	
3	2013	5	14	1841	-4	2122	-34	DL	N914DL	
4	2013	7	21	1102	-3	1230	-8	9E	N823AY	

	flight	origin	dest	air_time	distance	hour	minute
0	407	JFK	LAX	313	2475	9	40
1	329	JFK	SJU	216	1598	16	57
2	422	JFK	LAX	376	2475	8	59
3	2391	JFK	TPA	135	1005	18	41
4	3652	LGA	ORF	50	296	11	2

Filtro a través del valor de una columna

[ ]:
```
df1 = df[df["origin"]=="LGA"]
df1.head()
```

[ ]:
```
df1 = df[df["origin"]!="LGA"]
df1.head()
```

[ ]:
```
# & equivale a "AND"
df1 = df[(df["origin"]=="LGA") & (df["year"]==2013) & (df["air_time"]<=50)]
df1.head()
```

[ ]:
```
# | equivale a "OR"
df1 = df[(df["month"]==5) | (df["month"]==12)]
df1.head()
```

Es posible usar otras variables para filtrar datos:

[5]:
```
month_var = 5
df1 = df[(df["month"] == month_var) & (df["air_time"]>50)]
df1.head()
```

[5]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	\
1	2013	5	7	1657	-3	2104	10	DL	
3	2013	5	14	1841	-4	2122	-34	DL	
36	2013	5	13	1825	-4	2000	-35	DL	
63	2013	5	26	1417	-8	1529	-21	B6	
67	2013	5	25	1610	-5	1827	-13	MQ	

	tailnum	flight	origin	dest	air_time	distance	hour	minute
1	N3760C	329	JFK	SJU	216	1598	16	57
3	N914DL	2391	JFK	TPA	135	1005	18	41
36	N919DE	2131	LGA	DTW	77	502	18	25
63	N184JB	8	JFK	BUF	56	301	14	17
67	N525MQ	4657	LGA	ATL	102	762	16	10

Se puede negar cualquier criterio de filtro

[4]:
```
month_var = 5
df1 = df[~((df["month"] == month_var) & (df["air_time"]>50))]
df1.head()
```

[4]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	\
0	2013	6	30	940	15	1216	-4	VX	N626VA	
2	2013	12	8	859	-1	1238	11	DL	N712TW	
4	2013	7	21	1102	-3	1230	-8	9E	N823AY	
5	2013	1	1	1817	-3	2008	3	AA	N3AXAA	
6	2013	12	9	1259	14	1617	22	WN	N218WN	

	flight	origin	dest	air_time	distance	hour	minute
0	407	JFK	LAX	313	2475	9	40
2	422	JFK	LAX	376	2475	8	59
4	3652	LGA	ORF	50	296	11	2
5	353	LGA	ORD	138	733	18	17
6	1428	EWR	HOU	240	1411	12	59

Filtro a través de la función loc

[15]:
```
df1 = df.loc[df["origin"]=="LGA"]
df1.head()
```

[15]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
4	2013	7	21	1102	-3	1230	-8	9E	N823AY	3652	LGA	ORF	50	296	11	2
5	2013	1	1	1817	-3	2008	3	AA	N3AXAA	353	LGA	ORD	138	733	18	17
8	2013	9	26	725	-10	1027	-8	AA	N3FSAA	2279	LGA	MIA	148	1096	7	25
11	2013	11	22	1320	5	1628	-2	B6	N526JB	1639	LGA	RSW	161	1080	13	20
13	2013	3	25	2054	115	2256	91	FL	N919AT	645	LGA	ATL	104	762	20	54

[ ]:
```
df1 = df.loc[df["origin"]!="LGA"]
df1.head()
```

[ ]:
```
# & equivale a "AND"
df1 = df.loc[(df["origin"]=="LGA") & (df["year"]==2013) & (df["air_time"]<=50)]
df1.head()
```

[ ]:
```
# | equivale a "OR"
df1 = df.loc[(df["month"]==5) | (df["month"]==12)]
df1.head()
```

Es posible usar otras variables para filtrar datos

[ ]:
```
month_var = 5
df1 = df.loc[(df["month"] == month_var) & (df["air_time"]>50)]
df1.head()
```

Se puede negar cualquier criterio de filtro

[6]:
```
month_var = 5
df1 = df.loc[~((df["month"] == month_var) & (df["air_time"]>50))]
df1.head()
```

[6]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
0	2013	6	30	940	15	1216	-4	VX	N626VA	407	JFK	LAX	313	2475	9	40
2	2013	12	8	859	-1	1238	11	DL	N712TW	422	JFK	LAX	376	2475	8	59
4	2013	7	21	1102	-3	1230	-8	9E	N823AY	3652	LGA	ORF	50	296	11	2
5	2013	1	1	1817	-3	2008	3	AA	N3AXAA	353	LGA	ORD	138	733	18	17
6	2013	12	9	1259	14	1617	22	WN	N218WN	1639	LGA	RSW	161	1080	13	20

```
5     353    LGA  ORD       138       733    18      17
6    1428    EWR  HOU       240      1411    12      59
```

Filtro por posición de filas y columnas

[ ]:
```
df.iloc[:4] #First 4 rows, all columns
df.iloc[1:5,] #Second to fifth row
df.iloc[5,0] #Sixth row and first column
df.iloc[1:5,0] #Second to Fifth row, first column
df.iloc[1:5,:5] #Second to Fifth row, first 5 columns
df.iloc[2:7,1:3] #Third to Seventh row, 2nd and 3rd column
```

Filtro por posición de filas y nombre de columnas

[ ]:
```
# Selección de 5 filas y columnas con nombre "origin" y "distance"
df1 = df.loc[df.index[10:16],["origin","distance"]]
df1.head()
```

[ ]:
```
# Filtro múltiple
df1 = df.loc[(df["origin"]=="LGA") & (df["year"]==2013) & (df["air_time"]<=50),
             ["origin","distance","year"]]
df1.head()
```

Filtro para seleccionar múltiples valores

[3]:
```
df1 = df[df["origin"].isin(["JFK", "LGA"])]
df1.head()
```

[3]:
```
   year  month  day  dep_time  dep_delay  arr_time  arr_delay carrier tailnum  \
0  2013      6   30       940         15      1216         -4      VX  N626VA
1  2013      5    7      1657         -3      2104         10      DL  N3760C
2  2013     12    8       859         -1      1238         11      DL  N712TW
3  2013      5   14      1841         -4      2122        -34      DL  N914DL
4  2013      7   21      1102         -3      1230         -8      9E  N823AY

   flight origin dest  air_time  distance  hour  minute
0     407    JFK  LAX       313      2475     9      40
1     329    JFK  SJU       216      1598    16      57
2     422    JFK  LAX       376      2475     8      59
3    2391    JFK  TPA       135      1005    18      41
4    3652    LGA  ORF        50       296    11       2
```

Los valores que se deben conservar pueden ser almacenados en una lista

[ ]:
```
variables = ["JFK","LGA"]
df1 = df[df["origin"].isin(variables)]
df1.head()
```

Los valores que **no** se deben conservar pueden ser almacenados en una lista

[ ]:
```
remove_var = ["JFK","LGA"]
df1 = df[~df["origin"].isin(remove_var)]
df1.head()
```

Filtar columnas

[11]:
```
# Selección de columna: 'flight' hasta 'dest' para todas las filas
df1 = df.loc[:,"flight":"dest"]
df1.head()
```

[11]:

	flight	origin	dest
0	407	JFK	LAX
1	329	JFK	SJU
2	422	JFK	LAX
3	2391	JFK	TPA
4	3652	LGA	ORF

[9]:
```
# Selección de columna: 'flight' y 'dest'
df1 = df.loc[:,["flight","dest"]]
df1.head()
```

[9]:

	flight	dest
0	407	LAX
1	329	SJU
2	422	LAX
3	2391	TPA
4	3652	ORF

[8]:
```
# Selección de columna: 'flight' y 'dest'
df1 = df[["flight","dest"]]
df1.head()
```

[8]:

	flight	dest
0	407	LAX
1	329	SJU
2	422	LAX
3	2391	TPA
4	3652	ORF

[37]:
```
# Selección de columnas diferentes de 'flight', 'dest' y 'year'
# También es posible eliminar filas si fuese necesario
df1 =df.drop(['flight','dest','year'], axis = 1)
df1.head()
```

[37]:

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	\
0	6	30	940	15	1216	-4	VX	N626VA	
1	5	7	1657	-3	2104	10	DL	N3760C	
2	12	8	859	-1	1238	11	DL	N712TW	
3	5	14	1841	-4	2122	-34	DL	N914DL	

```
4      7   21      1102         -3      1230         -8      9E  N823AY
```

```
  origin  air_time  distance  hour  minute
0    JFK       313      2475     9      40
1    JFK       216      1598    16      57
2    JFK       376      2475     8      59
3    JFK       135      1005    18      41
4    LGA        50       296    11       2
```

FIltros con el método "query"

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

[4]:
```
df = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/
 ↪tips.csv")
df.head()
```

[4]:
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

Primero usamos el método .loc para visulaizar el resultado deseado

[5]:
```
mask = (df['sex']=='Male') & (df['size'] >2)
df.loc[mask].head()
```

[5]:
```
    total_bill   tip   sex smoker  day    time  size
1        10.34  1.66  Male     No  Sun  Dinner     3
2        21.01  3.50  Male     No  Sun  Dinner     3
5        25.29  4.71  Male     No  Sun  Dinner     4
7        26.88  3.12  Male     No  Sun  Dinner     4
13       18.43  3.00  Male     No  Sun  Dinner     4
```

Ahora mostramos el mismo resultado pero utilizando el método query. Nótese que los criterios de texto deben escribirse entre comillas dobles ""

[9]:
```
df.query('sex=="Male" & size > 2').head(5)
```

[9]:
```
    total_bill   tip   sex smoker  day    time  size
1        10.34  1.66  Male     No  Sun  Dinner     3
2        21.01  3.50  Male     No  Sun  Dinner     3
5        25.29  4.71  Male     No  Sun  Dinner     4
7        26.88  3.12  Male     No  Sun  Dinner     4
13       18.43  3.00  Male     No  Sun  Dinner     4
```

Es posible usar listas para múltiples criterios de filtro

[11]:
```
days = ["Sat", "Sun"]
df.query('sex=="Male" & size>2 & day == @days').head()
```

[11]:

	total_bill	tip	sex	smoker	day	time	size
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
5	25.29	4.71	Male	No	Sun	Dinner	4
7	26.88	3.12	Male	No	Sun	Dinner	4
13	18.43	3.00	Male	No	Sun	Dinner	4

Filtar valores duplicados Devuelve aquellos elementos que tienen valores duplicados. Así, la tabla resultante contendrá valores únicos según los parámetros de la función.

[86]:
```
# keep: If 'first', it considers first value as unique and rest of the same␣
 ↪values as duplicate.
# keep: If 'last', it considers last value as unique and rest of the same␣
 ↪values as duplicate.
# keep: If False, it consider all of the same values as duplicates
# inplace: Boolean values, removes rows with duplicates if True.
# ignore_indexbool, default False. If True, the resulting axis will be labeled␣
 ↪0, 1, …, n - 1.

df1 = df.drop_duplicates(subset = "origin",
                         keep = 'first',
                         inplace = False,
                         ignore_index = False)

df1.head()
```

[86]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	\
0	2013	6	30	940	15	1216	-4	VX	N626VA	
4	2013	7	21	1102	-3	1230	-8	9E	N823AY	
6	2013	12	9	1259	14	1617	22	WN	N218WN	

	flight	origin	dest	air_time	distance	hour	minute
0	407	JFK	LAX	313	2475	9	40
4	3652	LGA	ORF	50	296	11	2
6	1428	EWR	HOU	240	1411	12	59

Se pueden buscar valores duplicados en una combinación de varias columnas

[87]:
```
# keep: If 'first', it considers first value as unique and rest of the same␣
 ↪values as duplicate.
# keep: If 'last', it considers last value as unique and rest of the same␣
 ↪values as duplicate.
# keep: If False, it consider all of the same values as duplicates
# inplace: Boolean values, removes rows with duplicates if True.
```

```
# ignore_indexbool, default False. If True, the resulting axis will be labeled␣
 ↪0, 1, …, n - 1.

df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', 'Alejandro', 'Santiago'],
                   'Age'        : [23, 24, 24, 25],
                   'University' : ['AA', 'BB', 'BB', 'DD']})

df1 = df.drop_duplicates(subset = ["Age","University"],
                         keep = False,
                         inplace = False,
                         ignore_index = True)

df1.head()
```

[87]:

	Name	Age	University
0	Carlos	23	AA
1	Santiago	25	DD

Conservar **solamente** los valores duplicados

[95]:
```
# keep{'first', 'last', False}, default 'first'
# Determines which duplicates (if any) to mark.
# first : Mark duplicates as True except for the first occurrence.
# last : Mark duplicates as True except for the last occurrence.
# False : Mark all duplicates as True.

df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', 'Alejandro', 'Santiago'],
                   'Age'        : [23, 24, 24, 25],
                   'University' : ['AA', 'BB', 'BB', 'BB']})

df1 = df[df[["Age","University"]].duplicated(keep = False)]
df1.head()
```

[95]:

	Name	Age	University
1	Andrés	24	BB
2	Alejandro	24	BB

Filtrar valores faltantes

[50]:
```
# La librería "numpy" proporciona una forma para generar un valor vacío
import numpy as np

# axis
# axis{0 or 'index', 1 or 'columns'}, default 0
# Determine if rows or columns which contain missing values are removed.
# 0, or 'index' : Drop rows which contain missing values.
# 1, or 'columns' : Drop columns which contain missing value.

# how
```

```
# how: {‘any’, ‘all’}, default ‘any’
# Determine if row or column is removed from DataFrame, when we have at least␣
 ↪one NA or all NA.
# ‘any’ : If any NA values are present, drop that row or column.
# ‘all’ : If all values are NA, drop that row or column.

# subset
# subset: column label or sequence of labels, optional
# Labels along other axis to consider, e.g. if you are dropping rows these␣
 ↪would be a list of columns to include.

# inplace
# inplace: bool, default False
# If True, do operation inplace and return None.

# La función dropna y notnull pueden identificar a los valores de np.nan, None␣
 ↪y pd.NA

df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Santiago',␣
 ↪"Fernand0", "Marcelo", np.nan],
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})
#df1 = df.dropna(axis = 0, how = "all")
df1 = df.dropna(axis = 0, how = "any")
df1.head(10)
```

[50]:

	Name	Age	University
0	Carlos	23.0	AA
5	Marcelo	27.0	EE

Cuenta de los valores no nulos en el data frame

[30]:
```
column_notnull = df.notnull().sum()
column_notnull
```

[30]:
```
Name          5
Age           5
University    4
dtype: int64
```

Filtrar filas según una cadena de texto

[12]:
```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina',␣
 ↪"Fernand0", "Marcelo", np.nan],
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuya primera letra es 'C'
```

```
df1 = df[df['Name'].str[0] == 'C' ]
df1.head()
```

[12]:
```
      Name   Age University
0   Carlos  23.0         AA
3  Carmina  25.0       None
```

[15]:
```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina',␣
 ↪"Fernand0", "Marcelo", np.nan],
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuyas dos primeras letras es 'Ca'
df1 = df[df['Name'].str[0:2] == 'Ca' ]
df1.head()
```

[15]:
```
      Name   Age University
0   Carlos  23.0         AA
3  Carmina  25.0       None
```

[36]:
```
df['Name'].str[0:5]
```

[36]:
```
0    Carlo
1    André
2      NaN
3    Carmi
4    Ferna
5    Marce
6      NaN
Name: Name, dtype: object
```

[55]:
```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina',␣
 ↪"Fernand0", "Marcelo", np.nan],
                   'Age'        : [23, 24, 24, 25, None, 27, None],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuyas dos primeras letras es 'Ca'
df1 = df[df['Name'].str.startswith("Ca") == True]
df1.head()
```

[55]:
```
      Name   Age University
0   Carlos  23.0         AA
3  Carmina  25.0       None
```

[57]:
```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Carmina',␣
 ↪"Fernand0", "Marcos", np.nan],
                   'Age'        : [23, 24, 24, 25, None, 27, None],
```

```
                    'University' : ['AA', pd.NA, 'BB', None, 'CC','EE', pd.NA]})

# Filtrar filas de la columna 'Name' cuyas dos últimas letras es 'os'
df1 = df[df['Name'].str.endswith("os") == True]
df1.head()
```

[57]:
```
     Name   Age University
0  Carlos  23.0         AA
5  Marcos  27.0         EE
```

[51]:
```
x = df["Name"]
x.str.startswith("Ca")
```

[51]:
```
0     True
1    False
2      NaN
3     True
4    False
5    False
6      NaN
Name: Name, dtype: object
```