Actividad 2 - Preprocesamiento de datos

Raúl Correa Ocañas

A01722401

21-02-2024

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

In [ ]:
```
# Importa tus bases de datos como “data frame” (si usas Python o R) y compara el co
# de los dos files para ver si contienen la misma estructura de datos.
cars1 = pd.read_csv(r'C:\Users\Raul\OneDrive\Escritorio\CS\TC2004B.101\data\Cars1.c
cars2 = pd.read_csv(r'C:\Users\Raul\OneDrive\Escritorio\CS\TC2004B.101\data\Cars2.c
```

In [ ]:
```
# Si las dos bases de datos tienen la misma estructura, une las dos bases de datos
display(cars1.head())
display(cars2.head())
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin	car
0	18.0	8	307	130	3504	12	70	1	chevrolet chevelle malibu
1	15.0	8	350	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318	150	3436	11	70	1	plymouth satellite
3	16.0	8	304	150	3433	12	70	1	amc rebel sst
4	17.0	8	302	140	3449	10.5	70	1	ford torino

	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin	car
0	33.0	4	91	53	1795	17.4	76	3	honda civic
1	20.0	6	225	100	3651	17.7	76	1	dodge aspen se
2	18.0	6	250	78	3574	21	76	1	ford granada ghia
3	18.5	6	250	110	3645	16.2	76	1	pontiac ventura sj
4	17.5	6	258	95	3193	17.8	76	1	amc pacer d/l

In [ ]:
```
# Si las dos bases de datos tienen la misma estructura, une las dos bases de datos
print(cars1.shape)
print(cars2.shape)
```

```
(198, 11)
(200, 9)
```

Concluimos que su estructura de datos no es la misma. Cars1 tiene las columnas data1 y data2, mientras que Cars2 no las tiene. Por lo tanto, no podemos hacer un merge propio de las dos bases de datos."

In [ ]:
```
# En la base de datos aparecen columnas que contienen sólo valores NaN (Not a Numbe
cars1.drop(['data1','data2'], axis=1, inplace=True)
cars1
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin	
0	18.0	8	307	130	3504	12	70	1	chev che m
1	15.0	8	350	165	3693	11.5	70	1	sk
2	18.0	8	318	150	3436	11	70	1	plym sat
3	16.0	8	304	150	3433	12	70	1	reb
4	17.0	8	302	140	3449	10.5	70	1	t
...	...	...	...	...	...	...	...	...	
193	24.0	6	200	81	3012	17.6	76	1	mav
194	22.5	6	232	90	3085	17.6	76	1	h
195	29.0	4	85	52	2035	22.2	76	1	chev che
196	24.5	4	98	60	2164	22.1	76	1	chev w
197	29.0	4	90	70	1937	14.2	76	2	vw r

198 rows × 9 columns

In [ ]:
```
# Ahora las dos bases de datos tienen la misma estructura.
cars = pd.concat([cars1, cars2])
cars
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin	
0	18.0	8	307	130	3504	12	70	1	chev che m
1	15.0	8	350	165	3693	11.5	70	1	sk
2	18.0	8	318	150	3436	11	70	1	plym sat
3	16.0	8	304	150	3433	12	70	1	reb
4	17.0	8	302	140	3449	10.5	70	1	t
...	...	...	...	...	...	...	...	...	
195	27.0	4	140	86	2790	15.6	82	1	mus
196	44.0	4	97	52	2130	24.6	82	2	p
197	32.0	4	135	84	2295	11.6	82	1	d ram
198	28.0	4	120	79	2625	18.6	82	1	ra
199	31.0	4	119	82	2720	19.4	82	1	che

398 rows × 9 columns

In [ ]:
```
print("Columnas con valor ?")
for column in cars.columns:
    count = cars[column].isin(['?']).value_counts().get(True)
    print("{column}: {count}".format(column=column, count=count))
```

```
Columnas con valor ?
mpg: None
cylinders: None
displacement: None
horsepower: 9
weight: None
acceleration: 7
model: None
origin: None
car: None
```

In [ ]:
```
temp = cars.loc[cars['horsepower'] != '?', 'horsepower'].astype('int64')
cars.loc[cars['horsepower'] == '?', 'horsepower'] = temp.median()
cars['horsepower'] = cars['horsepower'].astype('int64')

temp = cars.loc[cars['acceleration'] != '?', 'acceleration'].astype('float64')
cars.loc[cars['acceleration'] == '?', 'acceleration'] = temp.mean()
cars['acceleration'] = cars['acceleration'].astype('float64')
```

In [ ]:
```
display(cars.loc[cars['horsepower'] == '?', 'horsepower'])
display(cars.loc[cars['acceleration'] == '?', 'acceleration'])
```

```
Series([], Name: horsepower, dtype: int64)
Series([], Name: acceleration, dtype: float64)
```

In [ ]:
```
print("Columnas con valor ?")
for column in cars.columns:
    count = cars[column].isin(['?']).value_counts().get(True)
    print("{column}: {count}".format(column=column, count=count))
```

```
Columnas con valor ?
mpg: None
cylinders: None
displacement: None
horsepower: None
weight: None
acceleration: None
model: None
origin: None
car: None
```

Hemos remplazado exitosaente los valores faltantes con la media y mediana de la columna correspondiente.

In [ ]:
```
# Muestra un resumen de los datos.
display(cars.info())
display(cars.describe())
display(cars.select_dtypes(include='object').describe())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 398 entries, 0 to 199
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    int64  
 3   horsepower    398 non-null    int64  
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model         398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car           398 non-null    object 
dtypes: float64(2), int64(6), object(1)
memory usage: 31.1+ KB
None
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	mo
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000
mean	23.514573	5.454774	193.427136	104.266332	2970.424623	15.550384	76.010
std	7.815984	1.701004	104.268683	38.225361	846.841774	2.731889	3.697
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000
25%	17.500000	4.000000	104.250000	76.000000	2223.750000	13.900000	73.000
50%	23.000000	4.000000	148.500000	93.000000	2803.500000	15.500000	76.000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000	17.000000	79.000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000

	car
count	398
unique	305
top	ford pinto
freq	6