Pandas Practice. Intelligent Systems

Jose Torres Postigo

2 de mayo de 2024

For this practice, we need to import the *pandas* library and the csv given (in my case, I'm importing it directly using its path in my machine):

```
import pandas as pd
df = pd.read_csv("/home/uni/intelligent-systems/labs/bmw.csv")
```

Excercise 1. *Show the first 10 samples of the dataset.*

Solution:

```
def f1():
    return df[0:10]
```

Output generated:

```
        model  year  price transmission  mileage fuelType  tax   mpg
    engineSize
0    5 Series  2014  11200    Automatic    67068   Diesel  125  57.6
          2.0
1    6 Series  2018  27000    Automatic    14827   Petrol  145  42.8
          2.0
2    5 Series  2016  16000    Automatic    62794   Diesel  160  51.4
          3.0
3    1 Series  2017  12750    Automatic    26676   Diesel  145  72.4
          1.5
4    7 Series  2014  14500    Automatic    39554   Diesel  160  50.4
          3.0
5    5 Series  2016  14900    Automatic    35309   Diesel  125  60.1
          2.0
6    5 Series  2017  16000    Automatic    38538   Diesel  125  60.1
          2.0
7    2 Series  2018  16250       Manual    10401   Petrol  145  52.3
          1.5
8    4 Series  2017  14250       Manual    42668   Diesel   30  62.8
          2.0
9    5 Series  2016  14250    Automatic    36099   Diesel   20  68.9
          2.0
```

Excercise 2. *Obtain the data series corresponding to the year atribute, and then obtain the data type and the number of samples of such series.*

Solution:

```
def f2():
    return df.get("year")
```

Output generated:

```
0         2014
1         2018
2         2016
3         2017
4         2014
          ...
10776     2016
10777     2016
10778     2017
10779     2014
10780     2017
Name: year, Length: 10781, dtype: int64
```

Excercise 3. *Obtain the data series corresponding to the mileage atribute, and then select the samples whose position in the series are multiples of 7.*

Solution:

```
def f3():
    x = df.get("mileage")
    return x[::7]
```

Output generated:

```
0        67068
7        10401
14       19057
21       78957
28       96213
         ...
10752    41500
10759    54008
10766    54987
10773    60372
10780    59432
Name: mileage, Length: 1541, dtype: int64
```

Excercise 4. *Obtain the data series corresponding to the mileage atribute, and then select randomly 40 % of the samples of the series.*

Solution:

```python
def f4():
    x = df.get("mileage")
    return x.sample(frac=0.4)
```

Output generated:

```
3381        17503
5936         5929
2679           12
3671        20506
8796         4971
            ...
2112        31238
5163        11055
10494       30183
8952        60018
9746        97600
Name: mileage, Length: 4312, dtype: int64
```

Excercise 5. *Obtain the data series corresponding to the mileage atribute, and then select the samples with a value lower than 20000 of that series.*

Solution:

```
def f5():
    x = df.get("mileage")
    return x[x < 20000]
```

Output generated:

```
1         14827
7         10401
14        19057
15        16570
39         6522
          ...
10740      3551
10741      2784
10742      5634
10743     13165
10755     13955
Name: mileage, Length: 5610, dtype: int64
```

Excercise 6. *Obtain the data series corresponding to the mpg atribute, and then sort the samples of the series.*

Solution:

```
def f6():
    x = df.get("mpg")
    return x.sort_values()
```

Output generated:

```
6965       5.5
6172       5.5
6132       5.5
6198       5.5
2116       5.5
         ...
7299     470.8
3628     470.8
6070     470.8
2352     470.8
7347     470.8
Name: mpg, Length: 10781, dtype: float64
```

Excercise 7. *Compute the mean, the standard deviation, the maximum and the minimum of the engineSize atribute*

Solution:

```
def f7():
    x = df.get("engineSize")
    return x.mean(), x.std(), x.max(), x.min()
```

Output generated:

```
(2.1677673685186902, 0.5520537772398375, 6.6, 0.0)
```

Excercise 8. *Obtain the number of rows and columns of the dataset, and the third sample starting from the end.*

Solution:

```python
def f8():
    return df.shape, df.iloc[-3]
```

Output generated:

```
((10781, 9),
model            3 Series
year                 2017
price               13100
transmission       Manual
mileage             25468
fuelType           Petrol
tax                   200
mpg                  42.8
engineSize            2.0
Name: 10778, dtype: object)
```

The tuple (10781, 9) indicates the number of rows and columns in that order.

Excercise 9. *Extract the mileage, price and mpg atributes to a new DataFrame, and then choose 20 % of the samples at random.*

Solution:

```
def f9():
    new_df = df[["mileage", "price", "mpg"]].copy()
    return new_df.sample(frac=0.2)
```

Output generated:

```
       mileage  price   mpg
4879     21388  12077  62.8
5183     29150  33990  47.1
9707     31910  22632  41.5
3199     21330  20980  44.8
4399      7726  25940  53.3
...        ...    ...   ...
10257    60187  13900  62.8
1396     36039  18010  58.9
6630      4186  35362  54.3
1846      1204  39950  33.6
9292     49636  11499  53.3

[2156 rows x 3 columns]
```

Excercise 10. *Obtain the samples whose value of the mileage atribute is lower than 10000 and the value of the mpg atribute is higher than 40.*

Solution:

```
def f10():
    return df[(df.get("mileage") < 10000) & (df.get("mpg") > 40)]
```

Output generated:

```
            model  year  price transmission  mileage fuelType  tax
   mpg  engineSize
131      1 Series  2017  14600    Automatic     5615   Petrol  145
   58.9          1.5
148      1 Series  2016  13700       Manual     8719   Petrol  125
   52.3          1.5
153      1 Series  2016  13750    Automatic     8707   Petrol   30
   55.5          1.5
166            X1  2020  31498    Semi-Auto     1560   Diesel  145
   60.1          2.0
167      2 Series  2020  27998       Manual     1580   Petrol  150
   43.5          1.5
...           ...   ...    ...          ...      ...      ...  ...
   ...         ...
10713    3 Series  2020  23899    Automatic     1255   Petrol  150
   47.9          2.0
10739    3 Series  2019  23987    Automatic     1049   Petrol  150
   47.9          2.0
10740    3 Series  2019  23454    Automatic     3551   Petrol  150
   47.9          2.0
10741    3 Series  2019  23599    Automatic     2784   Petrol  145
   47.9          2.0
10742    3 Series  2019  23499    Automatic     5634   Petrol  145
   47.9          2.0

[3079 rows x 9 columns]
```

Excercise 11. *Modify the values of the model atribute so that the "x Series"values are changed to "Series x", where x is a number between 1 and 9.*

Solution:

```
def f11():
    df['model'] = df['model'].replace(r'(\d) Series',
                                      r'Series \1',
                                      regex=True)
    return df
```

Output generated:

```
           model  year  price transmission  mileage fuelType  tax
    mpg  engineSize
0        Series 5  2014  11200    Automatic    67068   Diesel  125
    57.6          2.0
1        Series 6  2018  27000    Automatic    14827   Petrol  145
    42.8          2.0
2        Series 5  2016  16000    Automatic    62794   Diesel  160
    51.4          3.0
3        Series 1  2017  12750    Automatic    26676   Diesel  145
    72.4          1.5
4        Series 7  2014  14500    Automatic    39554   Diesel  160
    50.4          3.0
...           ...   ...    ...          ...      ...      ...  ...
    ...          ...
10776          X3  2016  19000    Automatic    40818   Diesel  150
    54.3          2.0
10777    Series 5  2016  14600    Automatic    42947   Diesel  125
    60.1          2.0
10778    Series 3  2017  13100       Manual    25468   Petrol  200
    42.8          2.0
10779    Series 1  2014   9930    Automatic    45000   Diesel   30
    64.2          2.0
10780          X1  2017  15981    Automatic    59432   Diesel  125
    57.6          2.0

[10781 rows x 9 columns]
```

Excercise 12. *Insert a new sample with the following values: model=“ 3 Series", year=2023, price = 22572, transmission = “Automatic", mileage = 74120, fuelType = “Diesel", tax = 160, mpg = 58.4, engineSize = 2.0*

Solution:

```
def f12():
    new_df = pd.DataFrame({
        "model": ["3 Series"],
        "year": [2023],
        "price": [22572],
        "transmission": ["Automatic"],
        "mileage": [74120],
        "fuelType": ["Diesel"],
        "tax": [160],
        "mpg": [58.4],
        "engineSize": [2.0],
    })
    return pd.concat([df, new_df], ignore_index=True)
    # ignore_index=True ignores overlapping indexes
```

Output generated:

```
             model  year  price transmission  mileage fuelType  tax
    mpg  engineSize
0        5 Series  2014  11200    Automatic    67068   Diesel  125
    57.6         2.0
1        6 Series  2018  27000    Automatic    14827   Petrol  145
    42.8         2.0
2        5 Series  2016  16000    Automatic    62794   Diesel  160
    51.4         3.0
3        1 Series  2017  12750    Automatic    26676   Diesel  145
    72.4         1.5
4        7 Series  2014  14500    Automatic    39554   Diesel  160
    50.4         3.0
...            ...   ...    ...          ...      ...      ...  ...
    ...         ...
10777    5 Series  2016  14600    Automatic    42947   Diesel  125
    60.1         2.0
10778    3 Series  2017  13100       Manual    25468   Petrol  200
    42.8         2.0
10779    1 Series  2014   9930    Automatic    45000   Diesel   30
    64.2         2.0
10780          X1  2017  15981    Automatic    59432   Diesel  125
    57.6         2.0
10781    3 Series  2023  22572    Automatic    74120   Diesel  160
    58.4         2.0

[10782 rows x 9 columns]
```

Excercise 13. *Convert the DataFrame into a numpy ndarray and print out the data type of the obtained ndarray.*

Solution:

```
def f13():
    return df.to_numpy()
```

Output generated:

```
[[' 5 Series' 2014 11200 ... 125 57.6 2.0]
 [' 6 Series' 2018 27000 ... 145 42.8 2.0]
 [' 5 Series' 2016 16000 ... 160 51.4 3.0]
 ...
 [' 3 Series' 2017 13100 ... 200 42.8 2.0]
 [' 1 Series' 2014 9930 ... 30 64.2 2.0]
 [' X1' 2017 15981 ... 125 57.6 2.0]]
```

Exercise 14. *Compute for each sample the average mileage per year.*

Solution:

```
def f14():
    return df["mileage"] / (2024 - df["year"])
```

Output generated:

```
0         6706.800000
1         2471.166667
2         7849.250000
3         3810.857143
4         3955.400000
             ...
10776     5102.250000
10777     5368.375000
10778     3638.285714
10779     4500.000000
10780     8490.285714
Length: 10781, dtype: float64
```