Outliers

August 8, 2023

1 Detecting and Replacing Outliers

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

[2]:
```
filename = os.path.join(os.getcwd(), "data", "adult.data.partial")
df = pd.read_csv(filename, header=0)
```

1.0.1 Get the Dimensions of the Dataset

[3]:
```
df.shape
```

[3]:
```
(7000, 15)
```

1.0.2 Glance at the Data

[4]:
```
df.head()
```

[4]:
```
   age  workclass  fnlwgt     education  education-num      marital-status  \
0   36  State-gov  112074     Doctorate             16       Never-married
1   35    Private   32528       HS-grad              9  Married-civ-spouse
2   21    Private  270043  Some-college             10       Never-married
3   45    Private  168837  Some-college             10  Married-civ-spouse
4   39    Private  297449     Bachelors             13  Married-civ-spouse

          occupation   relationship   race  sex_selfID  capital-gain  \
0    Prof-specialty  Not-in-family  White  Non-Female             0
1  Handlers-cleaners        Husband  White  Non-Female             0
2      Other-service      Own-child  White      Female             0
3       Adm-clerical           Wife  White      Female             0
4    Prof-specialty        Husband  White  Non-Female             0

   capital-loss  hours-per-week native-country  label
0             0              45  United-States  <=50K
1             0              45  United-States  <=50K
2             0              16  United-States  <=50K
```

```
3                0                24          Canada   >50K
4                0                40  United-States    >50K
```

1.1 Step 1: Compute the n-th Percentile of a Given Column

As an analyst, your goal is to detect the outliers in the `hours-per-week` column. In particular, you want to get the 99.9th percentile of the values in the `hours-per-week` column.

As was discussed in the videos, *z-scores* can be used to compute the n-th percentile of a data array. Toward the end of this notebook, we will be looking at a few ways to compute the z-scores and then figure out the n-th percentile in a data column. For now, however, we will show you a ready-made method from `numpy` that achieves our objective.

The code cell below uses the `np.percentile()` function and gets the value of `hours-per-week` that corresponds to the 99.9th percentile.

[5]:
```
hpw_999 = np.percentile(df['hours-per-week'], 99.9)
hpw_999
```

[5]:
```
99.0
```

In the code cell below, figure out the value of `education-num` that corresponds to the 90th percentile of the education in years. Hint: Use the same method as the code cell above, but replace the column name and the percentage value. Save your results to variable `edu_90`.

1.1.1 Graded Cell

The cell below will be graded. Remove the line "raise NotImplementedError()" before writing your code.

[7]:
```
# YOUR CODE HERE
edu_90=np.percentile(df['education-num'],90)
```

1.1.2 Self-Check

Run the cell below to test the correctness of your code above before submitting for grading. Do not add code or delete code in the cell.

[8]:
```
# Run this self-test cell to check your code;
# do not add code or delete code in this cell
from jn import testEdu

try:
    p, err = testEdu(df, edu_90)
    print(err)
except Exception as e:
    print("Error!\n" + str(e))
```

```
Correct!
```

1.2 Step 2: Add a Column With the Winsorized Version of the Original Column.

In this next section, we will use a new package called SciPy, which stands for Scientific Python. For more information about SciPy, consult the online documentation.

First, import the `stats` module from the `scipy` package.

[9]:
```
import scipy.stats as stats
```

Read the documentation for the function `stats.mstats.winsorize()`.

[10]:
```
stats.mstats.winsorize?
```

This function will create a copy of a given column, such that the outlier values will be replaced. In particular, you will pass two percentage cutoffs as a list to the `limits` parameter, and all the column values below the specified lower percentile cutoff, as well as all the values above the upper cutoff, will be replaced with the corresponding percentile value.

The code cell below uses the `stats.mstats.winsorize()` function to add a new column to DataFrame `df`. The column will be named `education-num-win` and will contain the winsorized version of the `education-num` column, with the cutoff from the 'bottom' and the cutoff from the 'top' both set at the 1% level.

[11]:
```
df['education-num-win'] = stats.mstats.winsorize(df['education-num'], limits=[0.
↪01, 0.01])
df.head(15)
```

[11]:

	age	workclass	fnlwgt	education	education-num	\
0	36	State-gov	112074	Doctorate	16	
1	35	Private	32528	HS-grad	9	
2	21	Private	270043	Some-college	10	
3	45	Private	168837	Some-college	10	
4	39	Private	297449	Bachelors	13	
5	27	Private	233421	Some-college	10	
6	40	Private	220460	HS-grad	9	
7	71	Private	163385	Some-college	10	
8	20	NaN	193416	Some-college	10	
9	41	Private	116391	Some-college	10	
10	54	Private	155408	HS-grad	9	
11	27	Self-emp-not-inc	140863	9th	5	
12	43	Private	170214	Some-college	10	
13	44	Private	198096	Bachelors	13	
14	32	Private	29933	Bachelors	13	

	marital-status	occupation	relationship	race	sex_selfID	\
0	Never-married	Prof-specialty	Not-in-family	White	Non-Female	
1	Married-civ-spouse	Handlers-cleaners	Husband	White	Non-Female	
2	Never-married	Other-service	Own-child	White	Female	
3	Married-civ-spouse	Adm-clerical	Wife	White	Female	
4	Married-civ-spouse	Prof-specialty	Husband	White	Non-Female	
5	Never-married	Adm-clerical	Own-child	White	Non-Female	
6	Never-married	Craft-repair	Not-in-family	White	Non-Female	
7	Widowed	Sales	Not-in-family	White	Non-Female	

```
8        Never-married                NaN      Own-child  White      Female
9   Married-civ-spouse       Craft-repair        Husband  White  Non-Female
10             Widowed  Handlers-cleaners      Unmarried  White      Female
11  Married-civ-spouse       Craft-repair        Husband  White  Non-Female
12  Married-civ-spouse    Exec-managerial        Husband  White  Non-Female
13  Married-civ-spouse              Sales        Husband  White  Non-Female
14  Married-civ-spouse  Handlers-cleaners        Husband  White  Non-Female

    capital-gain  capital-loss  hours-per-week native-country  label  \
0              0             0              45  United-States  <=50K
1              0             0              45  United-States  <=50K
2              0             0              16  United-States  <=50K
3              0             0              24         Canada   >50K
4              0             0              40  United-States   >50K
5              0             0              20  United-States  <=50K
6              0             0              40         Canada  <=50K
7              0             0              35  United-States   >50K
8              0             0              40  United-States  <=50K
9              0             0              40  United-States  <=50K
10             0             0              40  United-States  <=50K
11             0             0              40  United-States  <=50K
12             0             0              50  United-States   >50K
13          7688             0              40  United-States   >50K
14             0             0              50  United-States   >50K

    education-num-win
0                  16
1                   9
2                  10
3                  10
4                  13
5                  10
6                   9
7                  10
8                  10
9                  10
10                  9
11                  5
12                 10
13                 13
14                 13
```

1.3 Deep Dive: Computing z-scores

First, let's review what the *z-score of a given value* is. Say your dataset contains a feature (aka a one-dimensional array, a vector, a list, a variable, a data column) called `X`, and you want to compute the z-score for one particular observation (aka an example value, a cell) of this feature. Let's call

this observation x_i. A z-score of x_i is given by:

$$z = \frac{x_i - \bar{x}}{s},$$

where $\bar{x}$ is the mean of all the values of x in your data, and s is the standard deviation of those values.

The code cells below implements this formula.

Calculate a z-score for one (given) value, a given mean, and a given standard deviation

[12]:
```
F_mean = 5.44
F_std = 7.7
value = 4

value_zscore = (value-F_mean)/F_std
value_zscore
```

[12]:
```
-0.18701298701298705
```

Calculate a z-score for one (given) value, given the full sample of values. (The `numpy` way)

[13]:
```
F = [4, 6, 3, -3, 4, 5, 6, 7, 3 , 8, 1, 9, 1, 2, 2, 35, 4, 1]
value = F[0]

F_std = np.std(F)
F_mean = np.mean(F)
value_zscore = (value-F_mean)/F_std
value_zscore
```

[13]:
```
-0.1874826669747723
```

[14]:
```
F_mean
```

[14]:
```
5.444444444444445
```

Calculate the z-score for all values of a feature vector. (The `numpy` way) All we need to do now is to apply the computation we implemented above to every value in the feature vector `F`.

[15]:
```
F_std = np.std(F)
F_mean = np.mean(F)
zscores = []
for value in F:
    z = (value-F_mean)/F_std
    zscores.append(z)

zscores
```

[15]:
```
[-0.1874826669747723,
 0.07210871806722008,
 -0.3172783594957685,
 -1.0960525146217457,
```

```
-0.1874826669747723,
-0.057686974453776116,
0.07210871806722008,
0.2019044105882163,
-0.3172783594957685,
0.3317001031092125,
-0.5768697445377609,
0.46149579563020865,
-0.5768697445377609,
-0.4470740520167647,
-0.4470740520167647,
3.83618380117611,
-0.1874826669747723,
-0.5768697445377609]
```

Now, let's write code that implements the same computation the *pythonic* way -- using *list comprehensions*. Tip: remember that list comprehension syntax looks like this: `[action_to_apply(new_var_name) for new_var_name in list_containing_values]`

[16]:
```
F_std = np.std(F)
F_mean = np.mean(F)
zscores = [(value-F_mean)/F_std for value in F]
zscores
```

[16]:
```
[-0.1874826669747723,
 0.07210871806722008,
 -0.3172783594957685,
 -1.0960525146217457,
 -0.1874826669747723,
 -0.057686974453776116,
 0.07210871806722008,
 0.2019044105882163,
 -0.3172783594957685,
 0.3317001031092125,
 -0.5768697445377609,
 0.46149579563020865,
 -0.5768697445377609,
 -0.4470740520167647,
 -0.4470740520167647,
 3.83618380117611,
 -0.1874826669747723,
 -0.5768697445377609]
```

Calculate the z-score for all values of a feature vector. (The `scipy` way) Previously we were computing the z-score by implementing its definition formula via `numpy`. This time, we will use a ready-made function `zscore()` from the package `scipy`.

[17]:
```
zscores = stats.zscore(df['hours-per-week'])
zscores
```

[17]:
```
array([ 0.39704869,  0.39704869, -1.95626181, ..., -0.0086945 ,
       -0.0086945 ,  1.61427826])
```

Calculate z-scores for all values of all (numeric) columns We will demonstrate how to use the Pandas `apply()` method to broadcast the same function (`stats.zscore`) onto all columns in a (filtered!) DataFrame:

[18]:
```
df_zscores = df.select_dtypes(include=['number']).apply(stats.zscore)
df_zscores.head(10)
```

[18]:

	age	fnlwgt	education-num	capital-gain	capital-loss
0	-0.188926	-0.755763	2.305545	-0.153909	-0.212365
1	-0.261682	-1.503876	-0.406796	-0.153909	-0.212365
2	-1.280263	0.729900	-0.019319	-0.153909	-0.212365
3	0.465876	-0.221920	-0.019319	-0.153909	-0.212365
4	0.029341	0.987647	1.143113	-0.153909	-0.212365
5	-0.843728	0.385478	-0.019319	-0.153909	-0.212365
6	0.102097	0.263583	-0.406796	-0.153909	-0.212365
7	2.357526	-0.273195	-0.019319	-0.153909	-0.212365
8	-1.353018	0.009240	-0.019319	-0.153909	-0.212365
9	0.174853	-0.715163	-0.019319	-0.153909	-0.212365

	hours-per-week	education-num-win
0	0.397049	2.322443
1	0.397049	-0.413127
2	-1.956262	-0.022331
3	-1.307073	-0.022331
4	-0.008694	1.150056
5	-1.631667	-0.022331
6	-0.008694	-0.413127
7	-0.414438	-0.022331
8	-0.008694	-0.022331
9	-0.008694	-0.022331

[ ]:
```

```