DS_HW03

October 7, 2019

1 In this homework, I will preprocess some illegal data in my original dataset

2 Due the original dataset dosen't contain NaN, Null. So I randomly add some of these problematic conditions in original dataset

2.1 Start preprocess

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[34]:
```
raw_data=pd.read_csv('Automobile_data.csv')
print("The shape of this original dataset is {}".format(raw_data.shape))
# print(raw_data.head())
# print(raw_data.info())

print(raw_data.isnull().sum())
```

```
The shape of this original dataset is (205, 26)
symboling             0
normalized-losses     7
make                  0
fuel-type             0
aspiration            0
num-of-doors          5
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight          13
engine-type           0
```

```
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  0
stroke                0
compression-ratio     0
horsepower            0
peak-rpm              0
city-mpg              0
highway-mpg           0
price                 0
dtype: int64
```

3 Though the original dataset doesn't contain Null ,NaN. However it does contain quite a few '?' data, which means nothing useful as well.

3.1 Following I will replace the '?' data with Nan. So that I can easily locate them.

[35]:
```
raw_data=raw_data.replace('?',np.NaN)
print(raw_data.isnull().sum())
```

```
symboling             0
normalized-losses    48
make                  0
fuel-type             0
aspiration            0
num-of-doors          8
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           13
engine-type           0
num-of-cylinders      4
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
```

```
peak-rpm             2
city-mpg             0
highway-mpg          0
price                4
dtype: int64
```

4 And from the above analysis, we can find the features that needed to be modifeid.

4.1 {'normalized-losses', 'num-of-doors', 'curb-weight', 'num-of-clinders', 'bore', 'stroke' , 'horsepower' , 'peak-rpm' , 'price'}

Firstly, we deal with the normalized-losses

[40]:
```
wrong=raw_data[raw_data['normalized-losses']=='?']
empty=raw_data[raw_data['normalized-losses'].isnull()]
print(wrong.index)
print(empty.index)
```

```
Int64Index([], dtype='int64')
Int64Index([  0,   1,   2,   5,   7,   9,  14,  15,  16,  17,  43,  44,  45,
             46,  48,  49,  55,  63,  66,  71,  73,  74,  75,  82,  83,  84,
             99, 103, 109, 110, 113, 114, 124, 126, 127, 128, 129, 130, 131,
            139, 146, 154, 181, 189, 191, 192, 193, 203],
           dtype='int64')
```

We can firstly where are these null data. And then I choose to replace the empty data with the mean value of other non-empty data.

[41]:
```
a=raw_data[raw_data['normalized-losses'].notnull()]
b=(a['normalized-losses'].astype(int)).mean()
print(round(b))
raw_data['normalized-losses'].fillna(round(b),inplace=True)

empty=raw_data[raw_data['normalized-losses'].isnull()]
print(empty.index)
```

```
123.0
Int64Index([], dtype='int64')
```

4.2 ### And we can double check and find that there are no empty data in the ['normalized-losses']

4.2.1 Then we move on to the 'num-of-doors'

[42]:
```
print(raw_data['num-of-doors'])
print(raw_data[raw_data['num-of-doors'].isnull()].index)
```

```
0       two
1       two
2       two
3      four
4      four
       …
200    four
201    four
202    four
203    four
204    four
Name: num-of-doors, Length: 205, dtype: object
Int64Index([27, 53, 63, 92, 95, 99, 145, 173], dtype='int64')
```

[43]:
```
raw_data['num-of-doors'].fillna('four',inplace=True)
raw_data['num-of-doors']=raw_data['num-of-doors'].map({'two':2,'four':4})

print(raw_data['num-of-doors'])
print(raw_data[raw_data['num-of-doors'].isnull()].index)
```

```
0      2
1      2
2      2
3      4
4      4
      ..
200    4
201    4
202    4
203    4
204    4
Name: num-of-doors, Length: 205, dtype: int64
Int64Index([], dtype='int64')
```

4.2.2 Because most of the cars have four doors, so I decide to replace the null data with 'four'.

4.2.3 And then because it will be easier to use numeric value, so I map all the strings to numeric values.

4.2.4 Then move on to 'curb-weight'.

[45]:
```
print('Max : ',raw_data['curb-weight'].max())
print('Min : ',raw_data['curb-weight'].min())
print('Medain :',raw_data['curb-weight'].median())


print(raw_data[raw_data['curb-weight'].isnull()].index)
raw_data.fillna(raw_data['curb-weight'].median(),inplace=True)
print(raw_data[raw_data['curb-weight'].isnull()].index)
```

```
Max :  4066.0
Min :  1488.0
Medain : 2414.0
Int64Index([1, 7, 10, 22, 55, 63, 70, 103, 123, 136, 141, 144, 190],
dtype='int64')
Int64Index([], dtype='int64')
```

4.2.5 In this part, because the differenc between the maximum and minimum is rahter large. So I choose to use the medain to replace the empty value instead of mean value.

4.2.6 num-of-cylinder

[39]:
```
print(raw_data['num-of-cylinders'].value_counts())


print(raw_data[raw_data['num-of-cylinders'].isnull()].index)
raw_data['num-of-cylinders'].fillna('four',inplace=True)
print(raw_data[raw_data['num-of-cylinders'].isnull()].index)
```

```
four      156
six        24
five       10
eight       5
```

```
two         4
three       1
twelve      1
Name: num-of-cylinders, dtype: int64
Int64Index([9, 137, 144, 148], dtype='int64')
Int64Index([], dtype='int64')
```

4.3 ### We can find that the major number of cylinders is still 4, so I replace the empty data with four.

4.3.1 Bore and Stroke :

4.3.2 As for Bore and Stroke I choose to use mean value to replace the empty value.

[38]:
```
clean_by_mean=['bore','stroke']

for name in clean_by_mean:
    print(name, raw_data[raw_data[name].isnull()].index)
    raw_data[name].fillna((raw_data[name].astype(float)).mean(),inplace=True)
    print(name, raw_data[raw_data[name].isnull()].index)
```

```
bore Int64Index([55, 56, 57, 58], dtype='int64')
bore Int64Index([], dtype='int64')
stroke Int64Index([55, 56, 57, 58], dtype='int64')
stroke Int64Index([], dtype='int64')
```

4.3.3 Horsepower and peak-rpm :

4.3.4 As for Horsepower and peak-rpm I choose to use mean value to replace the empty value.

[37]:
```
clean_by_median=['horsepower','peak-rpm']


for name in clean_by_median:
    print(name, raw_data[raw_data[name].isnull()].index)
    raw_data[name].fillna((raw_data[name]).median(),inplace=True)
    print(name, raw_data[raw_data[name].isnull()].index)
```

```
horsepower Int64Index([130, 131], dtype='int64')
horsepower Int64Index([], dtype='int64')
peak-rpm Int64Index([130, 131], dtype='int64')
peak-rpm Int64Index([], dtype='int64')
```

4.3.5 Price :

4.3.6 As for price, we can observe the range is also very wide, hence the median will be more suitable to represent. So I replace the empty data with the median value.

[36]:
```
a=raw_data[raw_data['price'].notnull()]
b=a['price'].astype(int)
print('Max : ',b.max())
print('Min : ',b.min())
print('Mean : :',b.mean())
print('Median : ',b.median())

print(raw_data[raw_data['price'].isnull()].index)
raw_data['price'].fillna(b.median(),inplace=True)
print(raw_data[raw_data['price'].isnull()].index)
```

```
Max :  45400
Min :  5118
Mean : : 13207.129353233831
Median :  10295.0
Int64Index([9, 44, 45, 129], dtype='int64')
Int64Index([], dtype='int64')
```

4.4 Final check, there is no empty or null data in this dataset.

[46]:
```
print(raw_data.isnull().sum())
```

```
symboling            0
normalized-losses    0
make                 0
fuel-type            0
aspiration           0
num-of-doors         0
body-style           0
drive-wheels         0
engine-location      0
wheel-base           0
length               0
width                0
height               0
curb-weight          0
engine-type          0
num-of-cylinders     0
engine-size          0
fuel-system          0
```

```
bore                0
stroke              0
compression-ratio   0
horsepower          0
peak-rpm            0
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

[ ]: