21162101007_Kshitij_Gupta

21162101007_Kshitij_Gupta

1. Importing The DataSet

```
import pandas as pd
import numpy as np
```

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data"
df = pd.read_csv(path) #na_values are used to use NaN inplace of ?? in dataset. Nan is standard name for missing value
print("The first 5 rows of the dataframe")
headers = ["symboling","normalized-losses","make","fuel-type","aspiration", "num-of-doors","body-style",
         "drive-wheels","engine-location","wheel-base", "length","width","height","curb-weight","engine-type",
         "num-of-cylinders", "engine-size","fuel-system","bore","stroke","compression-ratio","horsepower",
         "peak-rpm","city-mpg","highway-mpg","price"]
#df.columns shows labels of columns in dataframe
df.columns = headers
df
```

```
The first 5 rows of the dataframe
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine locatic
0	3	?	alfa-romero	gas	std	two	convertible	rwd	fro
1	1	?	alfa-romero	gas	std	two	hatchback	rwd	fro
2	2	164	audi	gas	std	four	sedan	fwd	fro
3	2	164	audi	gas	std	four	sedan	4wd	fro
4	2	?	audi	gas	std	two	sedan	fwd	fro
...	...	...	...	...	...	...	...	...	
199	-1	95	volvo	gas	std	four	sedan	rwd	fro
200	-1	95	volvo	gas	turbo	four	sedan	rwd	fro
201	-1	95	volvo	gas	std	four	sedan	rwd	fro
202	-1	95	volvo	diesel	turbo	four	sedan	rwd	fro
203	-1	95	volvo	gas	turbo	four	sedan	rwd	fro

204 rows × 26 columns

2. Data Cleaning

```
# unique values of the column
for col in df.columns:
    print('{} => {}'.format(col, df[col].unique()))
```

```
stroke => ['2.68' '3.47' '3.40' '2.80' '3.19' '3.39' '3.03' '3.11' '3.23' '3.46'
 '3.90' '3.41' '3.07' '3.58' '4.17' '2.76' '3.15' '?' '3.16' '3.64' '3.10'
 '3.35' '3.12' '3.86' '3.29' '3.27' '3.52' '2.19' '3.21' '2.90' '2.07'
 '2.36' '2.64' '3.08' '3.50' '3.54' '2.87']
compression-ratio => [ 9.   10.    8.    8.5   8.3   7.    8.8   9.5   9.6   9.41  9.4   7.6
  9.2  10.1   9.1   8.1  11.5   8.6  22.7  22.   21.5   7.5  21.9   7.8
  8.4  21.    8.7   9.31  9.3   7.7  22.5  23.  ]
horsepower => ['111' '154' '102' '115' '110' '140' '160' '101' '121' '182' '48' '70'
 '68' '88' '145' '58' '76' '60' '86' '100' '78' '90' '176' '262' '135'
 '84' '64' '120' '72' '123' '155' '184' '175' '116' '69' '55' '97' '152'
 '200' '95' '142' '143' '207' '288' '?' '73' '82' '94' '62' '56' '112'
 '92' '161' '156' '52' '85' '114' '162' '134' '106']
peak-rpm => ['5000' '5500' '5800' '4250' '5400' '5100' '4800' '6000' '4750' '4650'
 '4200' '4350' '4500' '5200' '4150' '5600' '5900' '5750' '?' '5250' '4900'
 '4400' '6600' '5300']
city-mpg => [21 19 24 18 17 16 23 20 15 47 38 37 31 49 30 27 25 13 26 36 22 14 45 28
 32 35 34 29 33]
highway-mpg => [27 26 30 22 25 20 29 28 53 43 41 38 24 54 42 34 33 31 19 17 23 32 39 18
 16 37 50 36 47 46]
price => ['16500' '13950' '17450' '15250' '17710' '18920' '23875' '?' '16430'
 '16925' '20970' '21105' '24565' '30760' '41315' '36880' '5151' '6295'
 '6575' '5572' '6377' '7957' '6229' '6692' '7609' '8558' '8921' '12964'
 '6479' '6855' '5399' '6529' '7129' '7295' '7895' '9095' '8845' '10295'
 '12945' '10345' '6785' '11048' '32250' '35550' '36000' '5195' '6095'
 '6795' '6695' '7395' '10945' '11845' '13645' '15645' '8495' '10595'
 '10245' '10795' '11245' '18280' '18344' '25552' '28248' '28176' '31600'
 '34184' '35056' '40960' '45400' '16503' '5389' '6189' '6669' '7689'
 '9959' '8499' '12629' '14869' '14489' '6989' '8189' '9279' '5499' '7099'
 '6649' '6849' '7349' '7299' '7799' '7499' '7999' '8249' '8949' '9549'
 '13499' '14399' '17199' '19699' '18399' '11900' '13200' '12440' '13860'
 '15580' '16900' '16695' '17075' '16630' '17950' '18150' '12764' '22018'
 '32528' '34028' '37028' '9295' '9895' '11850' '12170' '15040' '15510'
 '18620' '5118' '7053' '7603' '7126' '7775' '9960' '9233' '11259' '7463'
 '10198' '8013' '11694' '5348' '6338' '6488' '6918' '7898' '8778' '6938'
 '7198' '7788' '7738' '8358' '9258' '8058' '8238' '9298' '9538' '8449'
 '9639' '9989' '11199' '11549' '17669' '8948' '10698' '9988' '10898'
 '11248' '16558' '15998' '15690' '15750' '7975' '7995' '8195' '9495'
 '9995' '11595' '9980' '13295' '13845' '12290' '12940' '13415' '15985'
 '16515' '18420' '18950' '16845' '19045' '21485' '22470' '22625']
```

```
# Replacing special character ? with null values
for col in df.columns:
    df[col].replace({'?' : np.nan}, inplace=True)
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front
1	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front
2	2	164	audi	gas	std	four	sedan	fwd	front
3	2	164	audi	gas	std	four	sedan	4wd	front
4	2	NaN	audi	gas	std	two	sedan	fwd	front

5 rows × 26 columns

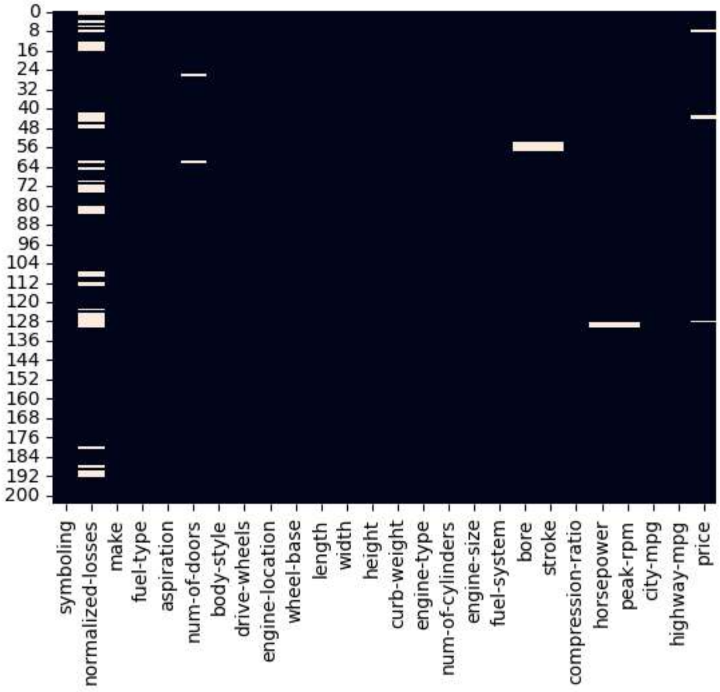
```
#getting count of all the null values for each column
df.isnull().sum()
```

```
symboling             0
normalized-losses    40
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
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                 4
stroke               4
compression-ratio    0
horsepower           2
peak-rpm             2
city-mpg             0
highway-mpg          0
price                4
dtype: int64
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df.isnull(),cbar=False)
```

```
<Axes: >
```



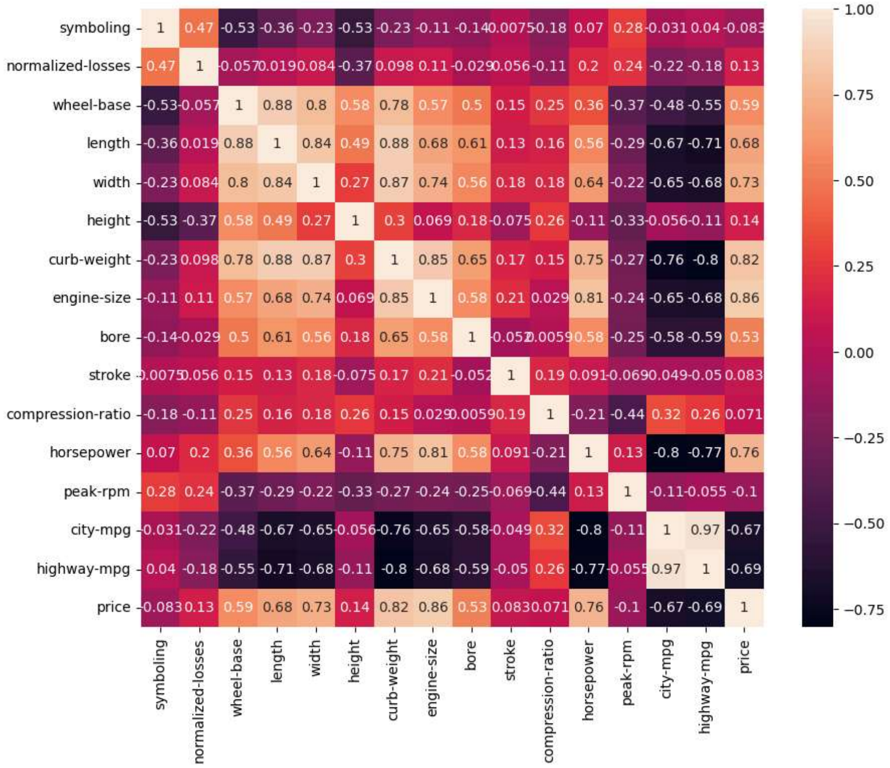
```
nullvalue_col = ['normalized-losses', 'bore', 'stroke', 'horsepower', 'peak-rpm', 'price']
for col in nullvalue_col:
    df[col] = pd.to_numeric(df[col])
    df[col].fillna(df[col].mean(), inplace=True)
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location
0	3	122.0	alfa-romero	gas	std	two	convertible	rwd	front
1	1	122.0	alfa-romero	gas	std	two	hatchback	rwd	front
2	2	164.0	audi	gas	std	four	sedan	fwd	front
3	2	164.0	audi	gas	std	four	sedan	4wd	front
4	2	122.0	audi	gas	std	two	sedan	fwd	front

5 rows × 26 columns

```
labelled = ['make','fuel-type','aspiration','num-of-doors','body-style','drive-wheels','engine-location','engine-type','fuel-system','num-of-
df1 = df.drop(columns=labelled)
fig, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(df1.corr(), cbar=True, annot=True,ax=ax)
```

```
<Axes: >
```

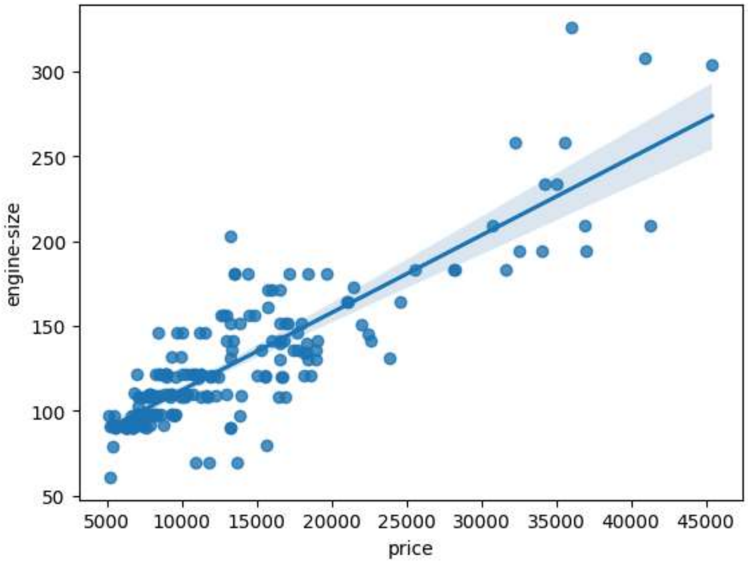


```
data_types = df.dtypes

# Identify the continuous attributes
continuous_attributes = data_types[data_types != 'object'].index.tolist()

# Print the list of continuous attributes
print(continuous_attributes)
print(len(continuous_attributes))
```

```
['symboling', 'normalized-losses', 'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-size', 'bore', 'stroke', 'compressi
16
```

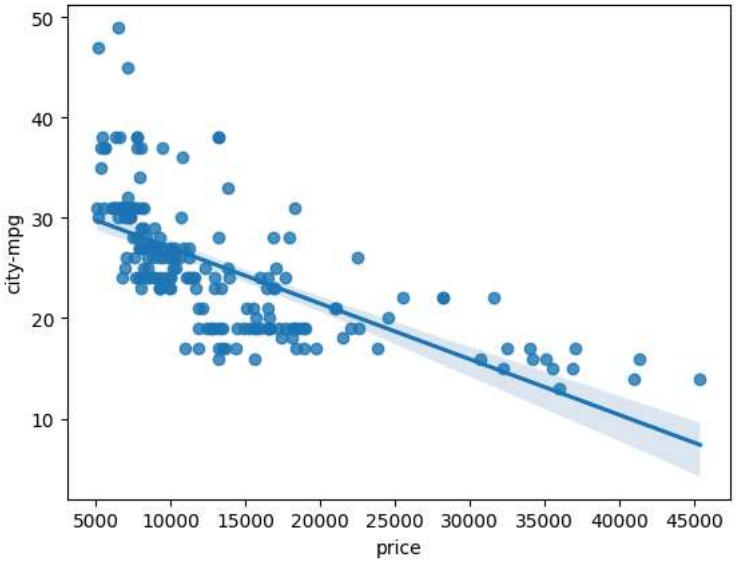
```
sns.regplot(data=df, x='price',y='engine-size')
```

```
<Axes: xlabel='price', ylabel='engine-size'>
```



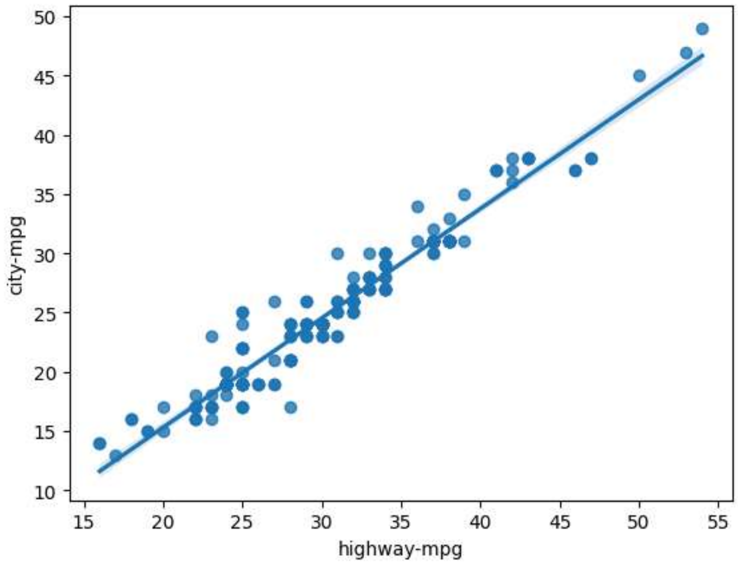
```
sns.regplot(data=df, x='price',y='city-mpg')
```

```
<Axes: xlabel='price', ylabel='city-mpg'>
```



```
sns.regplot(data=df, x='highway-mpg',y='city-mpg')
```

```
<Axes: xlabel='highway-mpg', ylabel='city-mpg'>
```



```
sns.regplot(data=df, x='price',y='horsepower')
```

```
<Axes: xlabel='price', ylabel='horsepower'>
```