jqu2vnjvw

January 23, 2025

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# to install not having library, pip insall seaborn
```

[ ]:
```
path='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/
↪automobileEDA.csv'
df=pd.read_csv(path)
df.head()
```

[ ]:
```
   symboling  normalized-losses         make aspiration num-of-doors  \
0          3                122  alfa-romero        std          two
1          3                122  alfa-romero        std          two
2          1                122  alfa-romero        std          two
3          2                164         audi        std         four
4          2                164         audi        std         four

    body-style drive-wheels engine-location  wheel-base    length  …  \
0  convertible          rwd           front        88.6  0.811148  …
1  convertible          rwd           front        88.6  0.811148  …
2    hatchback          rwd           front        94.5  0.822681  …
3        sedan          fwd           front        99.8  0.848630  …
4        sedan          4wd           front        99.4  0.848630  …

   compression-ratio  horsepower  peak-rpm city-mpg highway-mpg    price  \
0                9.0       111.0    5000.0       21          27  13495.0
1                9.0       111.0    5000.0       21          27  16500.0
2                9.0       154.0    5000.0       19          26  16500.0
3               10.0       102.0    5500.0       24          30  13950.0
4                8.0       115.0    5500.0       18          22  17450.0

  city-L/100km  horsepower-binned  diesel  gas
0    11.190476             Medium       0    1
1    11.190476             Medium       0    1
2    12.368421             Medium       0    1
```

```
3     9.791667             Medium       0    1
4    13.055556             Medium       0    1

[5 rows x 29 columns]
```

```
[ ]: df.shape #shape of
```

```
[ ]: (201, 29)
```

```
[ ]: df.isnull().sum()
```

```
[ ]: symboling            0
     normalized-losses    0
     make                 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               4
     compression-ratio    0
     horsepower           0
     peak-rpm             0
     city-mpg             0
     highway-mpg          0
     price                0
     city-L/100km         0
     horsepower-binned    1
     diesel               0
     gas                  0
     dtype: int64
```

```
[ ]: df['stroke'].value_counts()
```

```
[ ]: stroke
     3.40    19
     3.03    14
```

```
3.23    14
3.15    14
3.39    13
2.64    11
3.29     9
3.35     9
3.46     8
3.07     6
3.58     6
3.50     6
3.27     6
3.41     6
3.19     6
3.52     5
3.64     5
3.47     4
3.86     4
3.54     4
3.90     3
3.11     3
2.90     3
3.08     2
2.19     2
2.68     2
3.10     2
4.17     2
2.80     2
3.12     1
3.21     1
2.07     1
2.36     1
3.16     1
2.76     1
2.87     1
Name: count, dtype: int64
```

[ ]:
```python
stroke_mean=df['stroke'].mean()
stroke_mean
```

[ ]:
```
3.256903553299492
```

[ ]:
```python
df['stroke'].fillna('stroke_mean',inplace=True,limit=4)
```

```
<ipython-input-12-06d5d38756d7>:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
```

```
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['stroke'].fillna('stroke_mean',inplace=True,limit=4)
<ipython-input-12-06d5d38756d7>:1: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value 'stroke_mean' has dtype incompatible with float64, please
explicitly cast to a compatible dtype first.
  df['stroke'].fillna('stroke_mean',inplace=True,limit=4)
```

[ ]:
```
df['horsepower-binned'].value_counts()
```

[ ]:
```
horsepower-binned
Low       115
Medium     62
High       23
Name: count, dtype: int64
```

[ ]:
```
df['horsepower-binned'].fillna('Low',inplace=True,limit=1)
```

```
<ipython-input-15-bc4daa091fd1>:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['horsepower-binned'].fillna('Low',inplace=True,limit=1)
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
symboling            0
normalized-losses    0
make                 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
city-L/100km         0
horsepower-binned    0
diesel               0
gas                  0
dtype: int64
```

[ ]:
```
df['engine-size'].dtype #check the datatype of any column,row
```

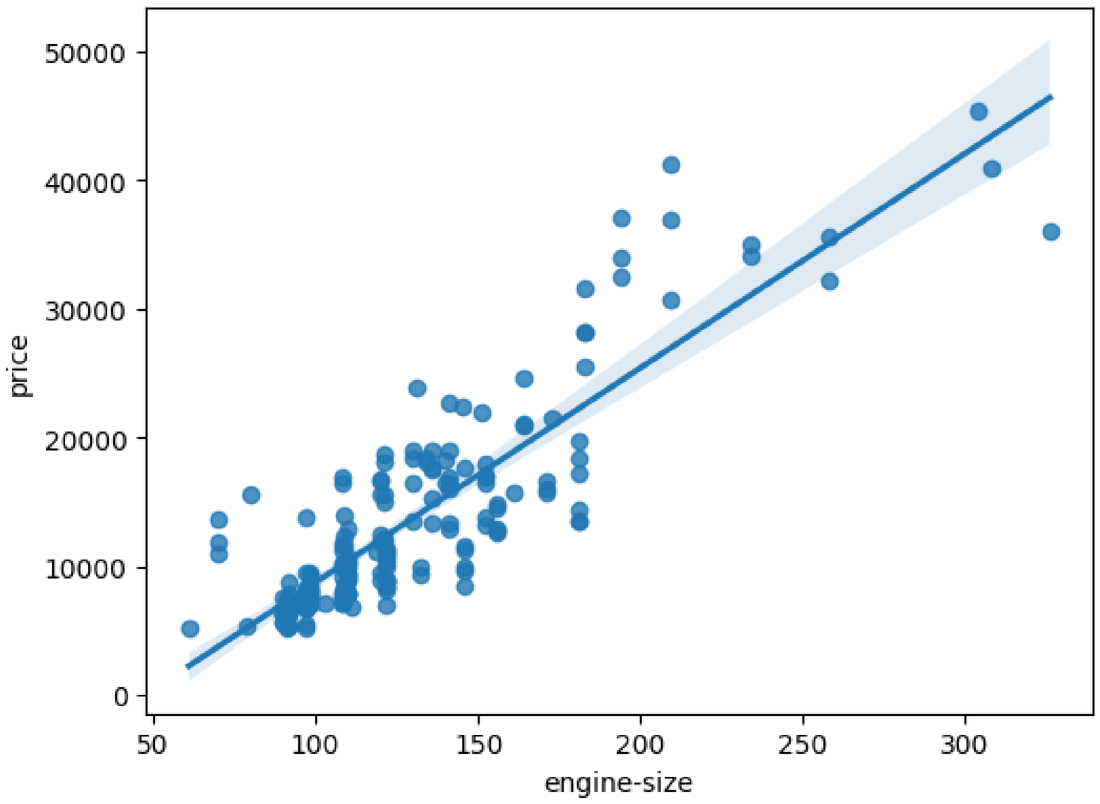
[ ]:
```
dtype('int64')
```

[ ]:
```
#correlation
df[['length','price','symbloing']].corr() #checks correlation of any variables
```

[ ]:
```
          length     price
length  1.000000  0.690628
price   0.690628  1.000000
```

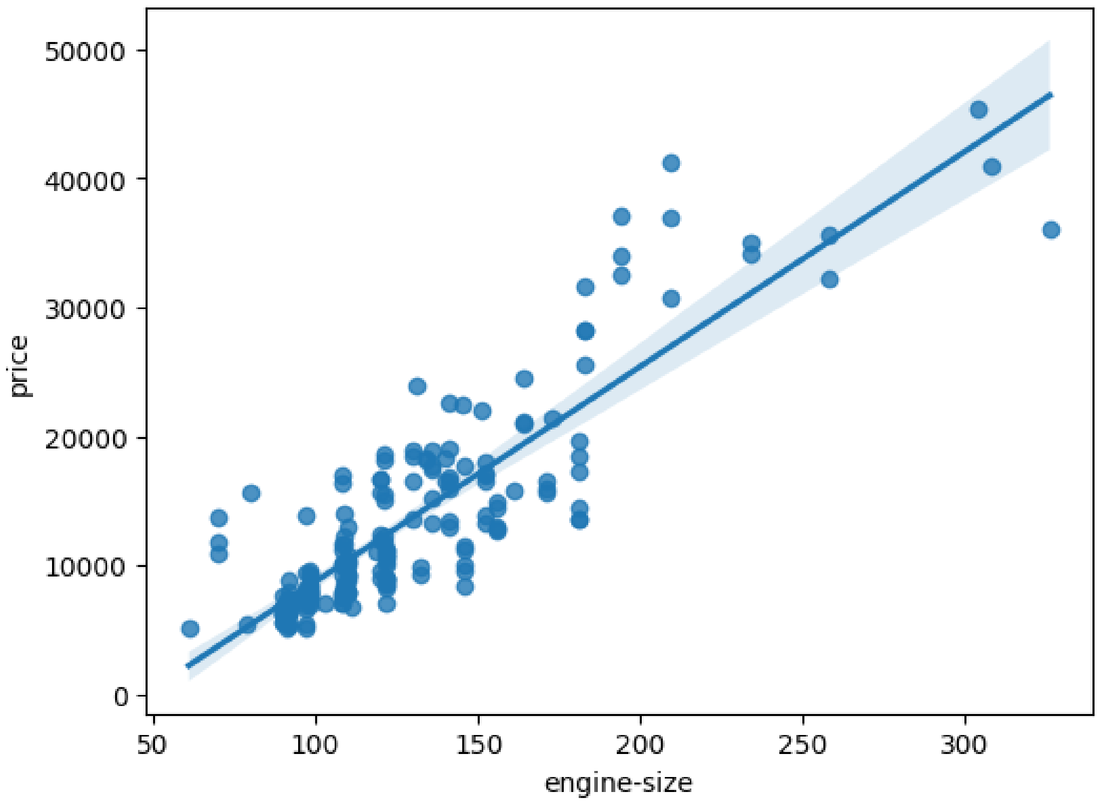
[ ]:
```
sns.regplot(x='engine-size',y='price',data=df)
```

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



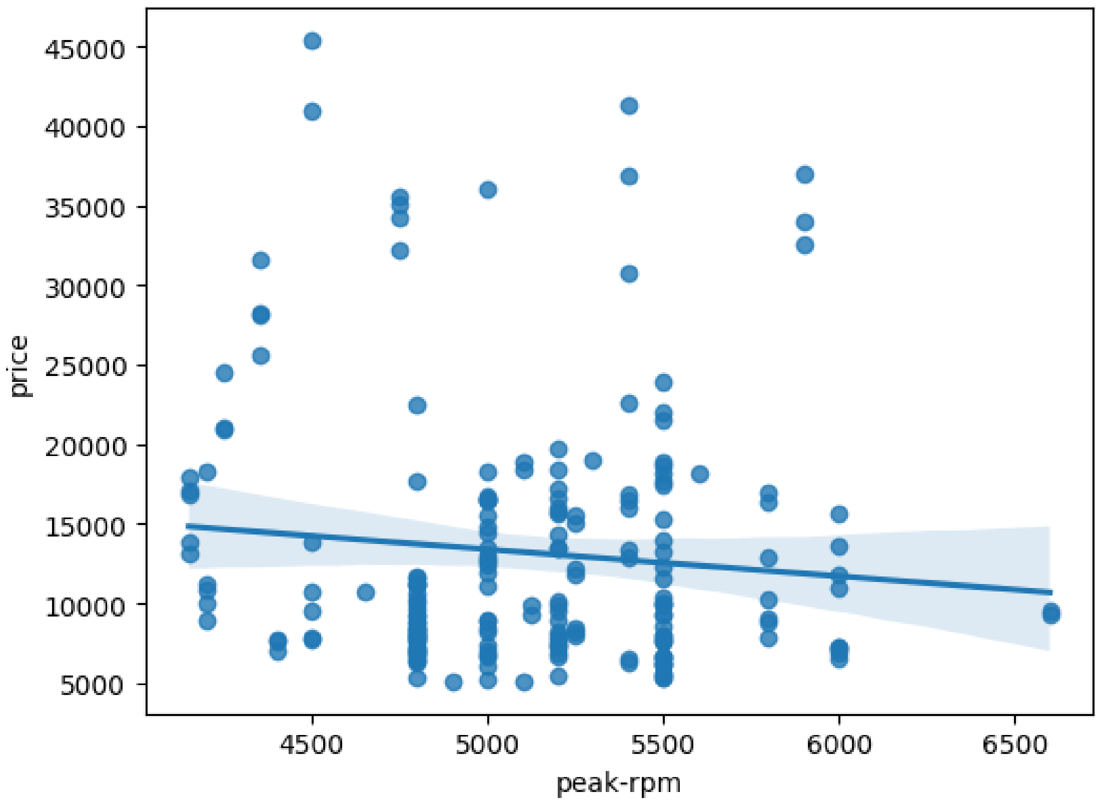
```
[ ]: sns.regplot(x=df['engine-size'],y=df['price'])
```

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



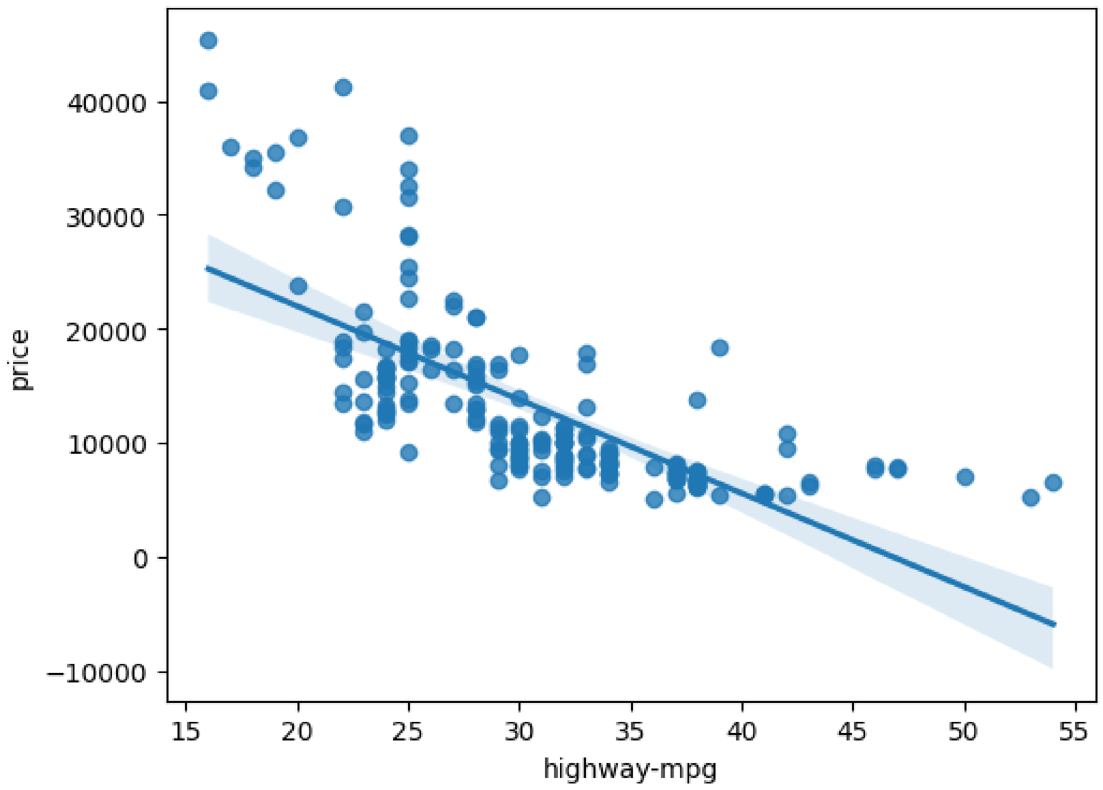
```
[ ]: sns.regplot(x=df['peak-rpm'],y=df['price'])
```

```
[ ]: <Axes: xlabel='peak-rpm', ylabel='price'>
```



```
[ ]: sns.regplot(x=df['highway-mpg'],y=df['price'])
```

```
[ ]: <Axes: xlabel='highway-mpg', ylabel='price'>
```

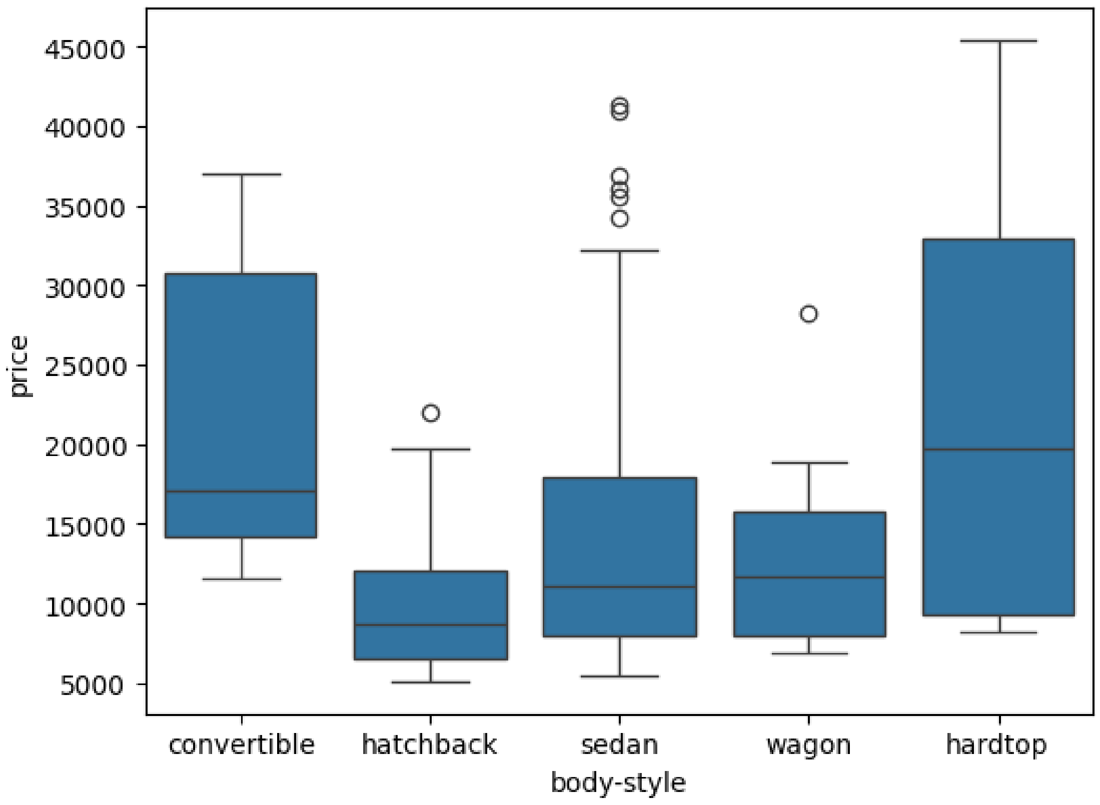


[ ]:
```
df['body-style'].value_counts()
```

[ ]:
```
body-style
sedan          94
hatchback      68
wagon          25
hardtop         8
convertible     6
Name: count, dtype: int64
```

[ ]:
```
#box plot
sns.boxplot(x=df['body-style'],y=df['price'])
```

[ ]:
```
<Axes: xlabel='body-style', ylabel='price'>
```



```
[ ]: df['drive-wheels'].value_counts()
```

```
[ ]: drive-wheels
     fwd    118
     rwd     75
     4wd      8
     Name: count, dtype: int64
```

```
[ ]: df['drive-wheels'].unique() #checks for unique values
```

```
[ ]: array(['rwd', 'fwd', '4wd'], dtype=object)
```

```
[ ]:
```