In [479…
```
import seaborn as sns
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import numpy as np
```

In [480…
```
data=pd.read_csv("D:\Kaggle\Linear Regression RFE(Recursive feature eimination)- CAR pr
data.head()  #prints first 5 rows
#print(data.head())
print(data.shape,data.size,data.ndim)
#data.shape() will raise error
#data.to_numpy()
```

```
(205, 26) 5330 2
```

In [ ]:

In [481…
```
#print(data.describe()) # gives mean, stdard deviation ,min,25percentile,50 percentile,
# it excludes the cloumns which contains the string object.

#data.describe(percentiles=[0.2,0.4,0.6,.8])   we can give percnetiles as our wish
include=['float','int','object']
data.describe(percentiles=[0.2,0.4,0.6,0.8],include='all') #returns NA for string objec
```

Out[481…

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	eng
count	205.000000	205.000000	205	205	205	205	205	205	
unique	NaN	NaN	147	2	2	2	5	3	
top	NaN	NaN	peugeot 504	gas	std	four	sedan	fwd	
freq	NaN	NaN	6	185	168	115	96	120	
mean	103.000000	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	
std	59.322565	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	
min	1.000000	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	
20%	41.800000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	
40%	82.600000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	
50%	103.000000	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	
60%	123.400000	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	
80%	164.200000	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	
max	205.000000	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	

13 rows × 26 columns

In [482…
```
data.describe(include='object')
```

Out[482…

	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype

	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype
count	205	205	205	205	205	205	205	205
unique	147	2	2	2	5	3	2	7
top	peugeot 504	gas	std	four	sedan	fwd	front	ohc
freq	6	185	168	115	96	120	202	148

In [483…
```
data.info()
#data.info(verbose=False) # verbose is used for whether t print full summary or not. fa
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

Data Cleaning and Preparation

In [484…
```
#Splitting company name from car name column
company_name=data['CarName'].apply(lambda x:x.split(' ')[0]) #Split method retuens a li
#breaking the given string by secified separator, here we used space as a separator
#company_name.dtype   data type is object i.e string
#print(company_name)
#data
#company_name
#company_name.insert(3,"CompanyName",company_name,allow_duplicates=False)    # error  s
data.insert(3,"CompanyName",company_name,allow_duplicates=True)
```

```
data.drop(labels=['CarName'],axis=1,inplace=True)  #axis=1 means column and 0 means row
data
```

Out[484...

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	eng
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	
...	...	...	...	...	...	...	...	...	
200	201	-1	volvo	gas	std	four	sedan	rwd	
201	202	-1	volvo	gas	turbo	four	sedan	rwd	
202	203	-1	volvo	gas	std	four	sedan	rwd	
203	204	-1	volvo	diesel	turbo	four	sedan	rwd	
204	205	-1	volvo	gas	turbo	four	sedan	rwd	

205 rows × 26 columns

In [485...

```
data.head()
```

Out[485...

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engin
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	

5 rows × 26 columns

In [486...

```
data.CompanyName.unique() #returns unique values of data in CompanyName column
```

Out[486...

```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'maxda', 'mazda', 'buick', 'mercury',
       'mitsubishi', 'Nissan', 'nissan', 'peugeot', 'plymouth', 'porsche',
       'porcshce', 'renault', 'saab', 'subaru', 'toyota', 'toyouta',
       'vokswagen', 'volkswagen', 'vw', 'volvo'], dtype=object)
```

In [487...

```
data.CompanyName.unique().size
```

Out[487... 28

Fixing invalid values

maxda=mazda nissan=Nissan porsche=porcshce toyota=toyouta vokswagen=volkswagen=vw

In [488…
```
data.CompanyName=data.CompanyName.str.lower() # converts all letters in CompanyName col
data.CompanyName.unique()
```

Out[488…
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'maxda', 'mazda', 'buick', 'mercury',
       'mitsubishi', 'nissan', 'peugeot', 'plymouth', 'porsche',
       'porcshce', 'renault', 'saab', 'subaru', 'toyota', 'toyouta',
       'vokswagen', 'volkswagen', 'vw', 'volvo'], dtype=object)
```

In [489…
```
data.CompanyName.replace("maxda","mazda",inplace=True)
#data.CompanyName.str.replace("Nissan","nissan")
data.CompanyName.replace("porcshce","porsche",inplace=True)
data.CompanyName.replace("toyouta","toyota",inplace=True)
data.CompanyName.replace("vokswagen","volkswagen",inplace=True)
data.CompanyName.replace("vw","volkswagen",inplace=True)
```

In [490…
```
data.CompanyName[150]
```

Out[490…
```
'toyota'
```

In [491…
```
data.CompanyName.unique()
```

Out[491…
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'nissan', 'peugeot', 'plymouth', 'porsche', 'renault', 'saab',
       'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

In [492…
```
data.loc[data.duplicated()]  #checking for duplicates by name of columns
```

Out[492…

car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	engineloc

0 rows × 26 columns

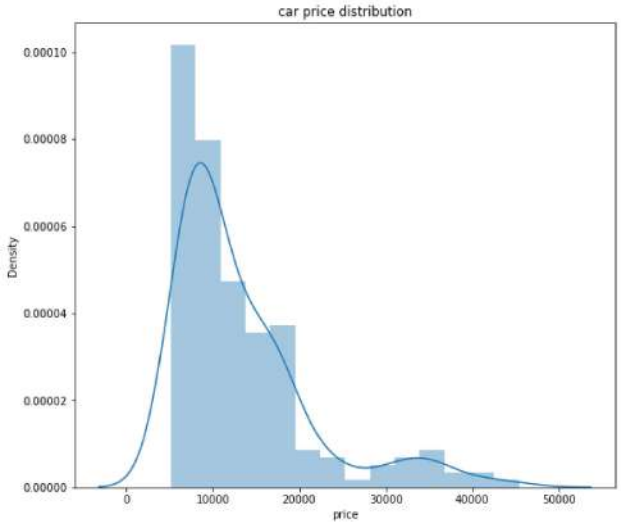
In [493…
```
data.columns
```

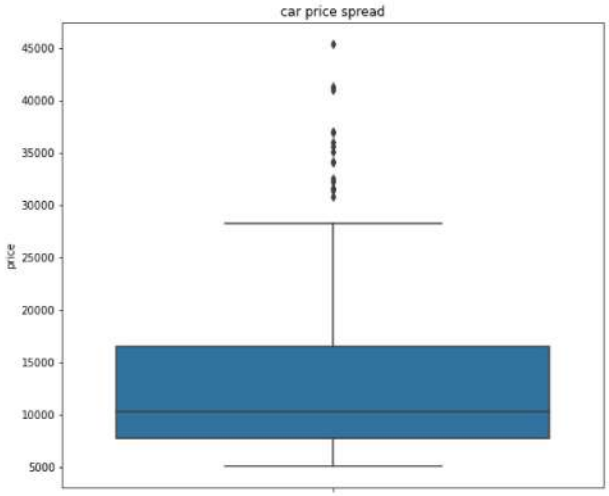
Out[493…
```
Index(['car_ID', 'symboling', 'CompanyName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

Visualizing the data

In [494…
```
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
plt.title("car price distribution")
sns.distplot(data.price)
#plt.show()
plt.subplot(1,2,2)  # will come in same row
plt.title("car price spread")
sns.boxplot(y=data.price)
plt.show()
```

```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```





In [495…
```
#data[price].describe(percentiles=[0.25,0.5,0.75,0.85,0.9,1]) # raises error
data.price.describe(percentiles=[0.25,0.5,0.75,0.85,0.9,1])
```

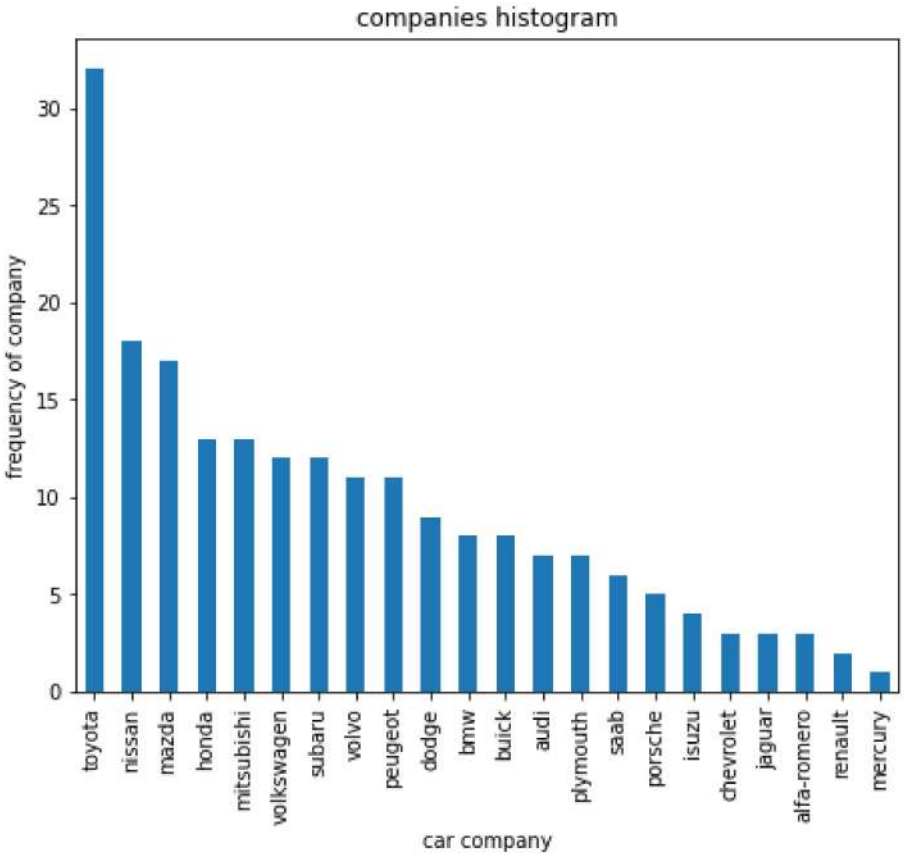
Out[495…
```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
25%       7788.000000
50%      10295.000000
75%      16503.000000
85%      18500.000000
90%      22563.000000
100%     45400.000000
max      45400.000000
Name: price, dtype: float64
```

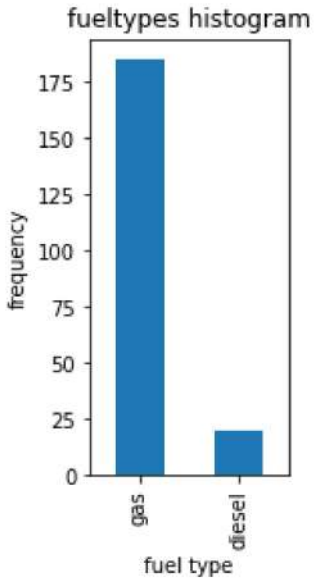
3.1 Visualizing Categorical Data

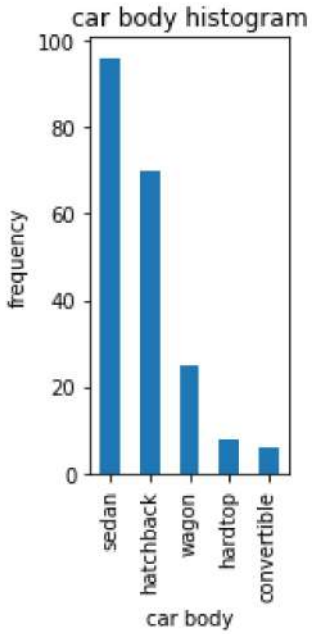
- CompanyName
- Symboling
- fueltype
- enginetype
- carbody
- doornumber
- enginelocation
- fuelsystem
- cylindernumber
- aspiration
- drivewheel

In [496…
```
plt.figure(figsize=(25,6))
plt.subplot(1,3,1)
```

```
plt1=data.CompanyName.value_counts().plot(kind='bar')  #returns count of unique values
plt1.set(xlabel="car company", ylabel="frequency of company")
plt.title("companies histogram")
plt.show()

plt.subplot(1,3,2)
plt2=data.fueltype.value_counts().plot(kind="bar")
plt2.set(xlabel="fuel type",ylabel="frequency")
plt.title("fueltypes histogram")
plt.show()

plt.subplot(1,3,3)
plt3=data.carbody.value_counts().plot(kind="bar")
plt3.set(xlabel="car body",ylabel="frequency") #plt3.title(xlabel="car body",ylabel="fr
plt.title("car body histogram")
plt.show()
```







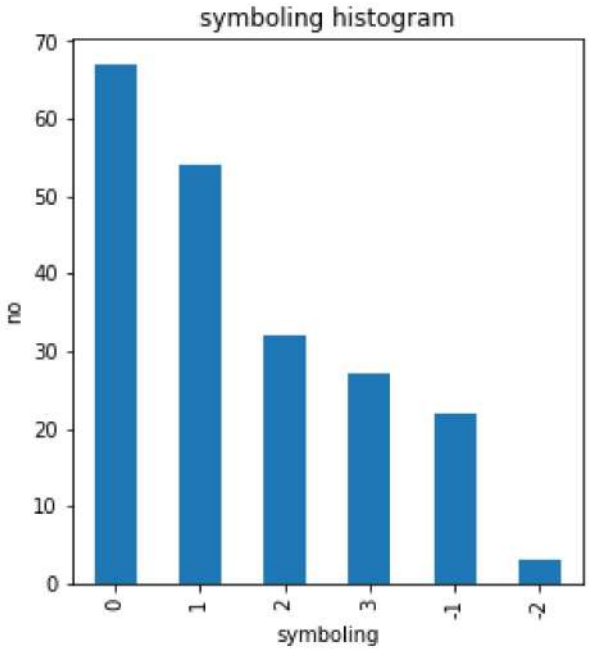
Inference for above

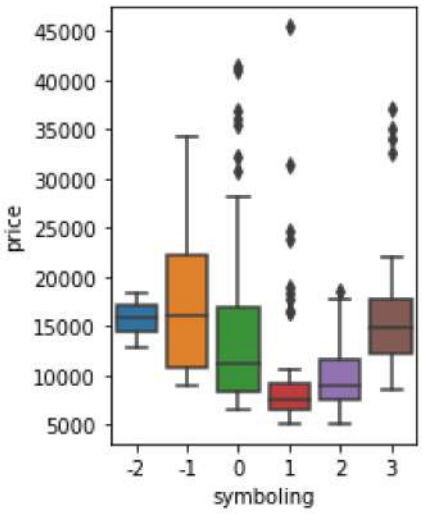
- toyota is most favouered comapny
- gas is used by most
- sedan is the most frquent car body

In [497…

```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt4=data.symboling.value_counts().plot(kind="bar")
plt4.set(xlabel="symboling", ylabel="no")
plt.title("symboling histogram")
plt.show()

plt.subplot(1,2,2)
#plt4.set(xlabel="symboling",ylabel="price")
#plt.title("symboling vs price")
```

```
sns.boxplot(x=data.symboling,y=data.price)
plt.show()
```





Inference for above

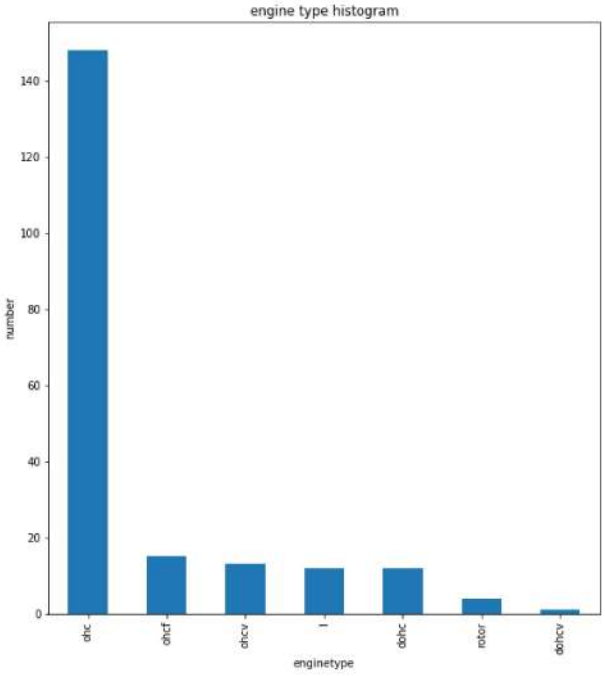
- Symboling with 1 and 0 have more no.of rows
- symboling with -2 have least no.of rows.
- the price is more for symboling -1..the price is least for symboling 1
- symboling with 3 has price range similar to symboling with -2

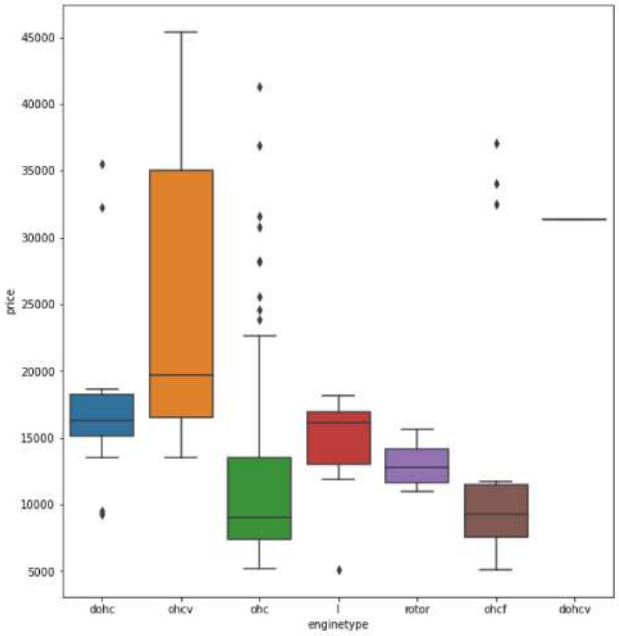
In [498…

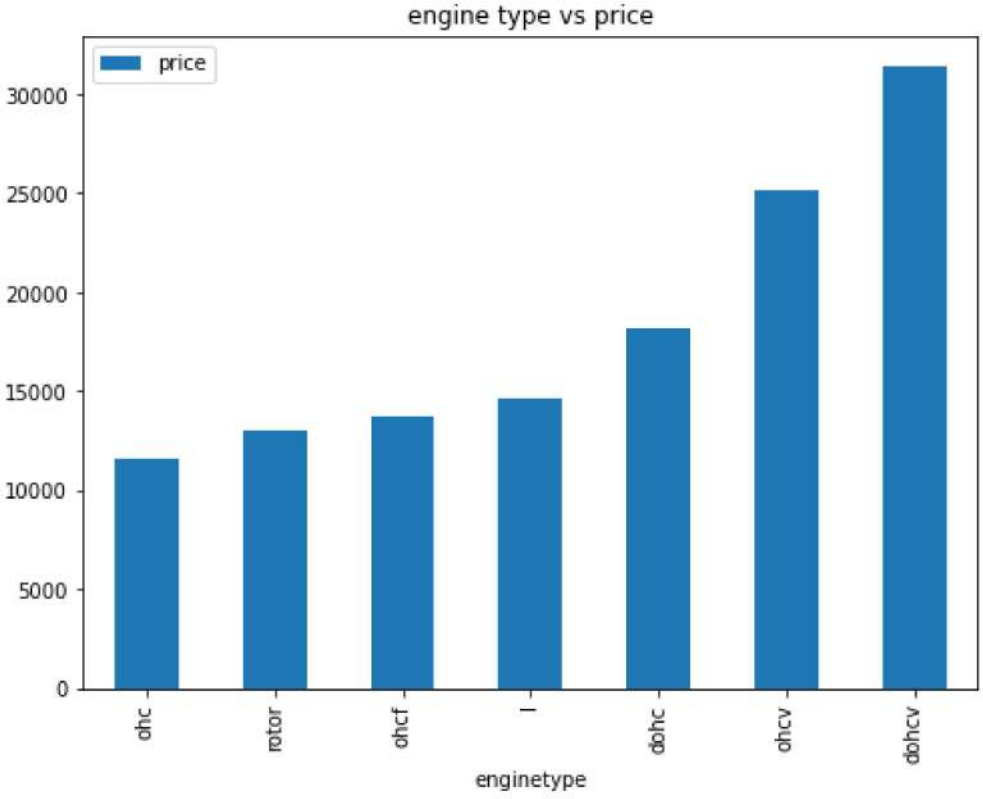
```
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.title("engine type histogram")
plt5=data.enginetype.value_counts().plot(kind="bar")
plt5.set(xlabel="enginetype",ylabel="number")
plt.subplot(1,2,2)
sns.boxplot(data.enginetype,data.price)
plt.show()
```

```
plt6=pd.DataFrame(data.groupby(['enginetype'])["price"].mean().sort_values())
(plt6)
plt6.plot.bar(figsize=(8,6))
plt.title("engine type vs price")
plt.show()
```

```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```







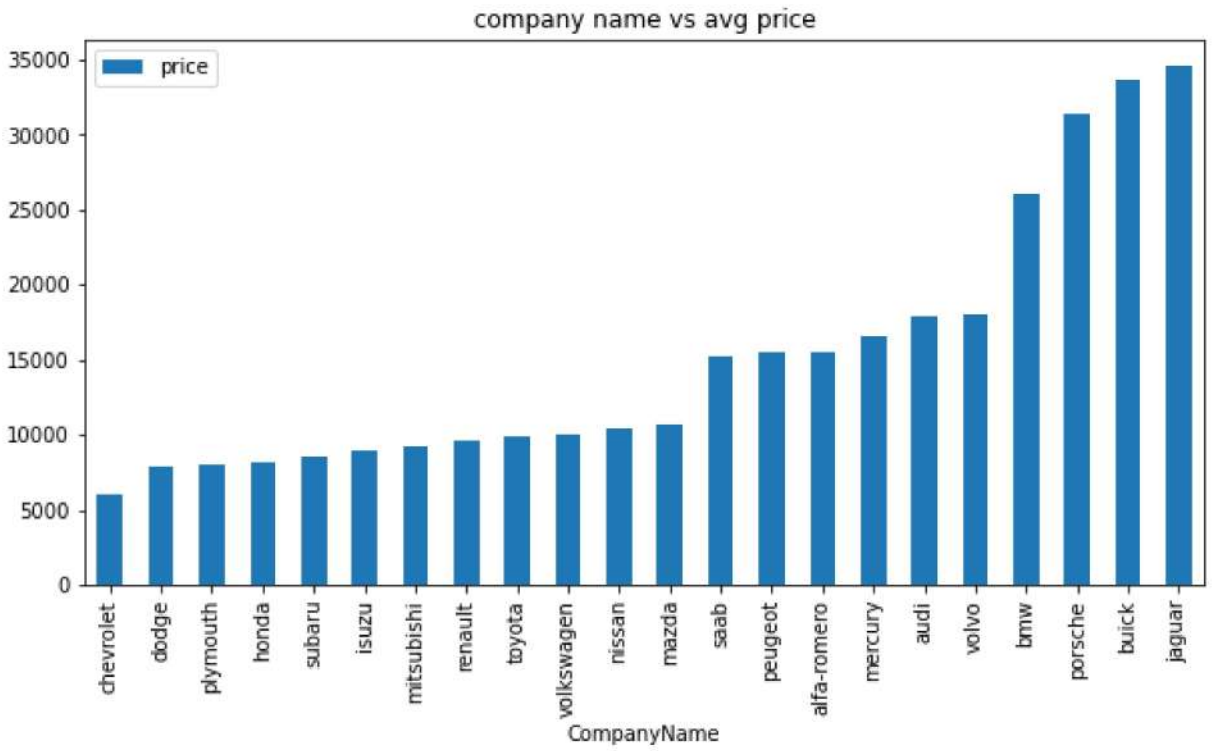
Inference

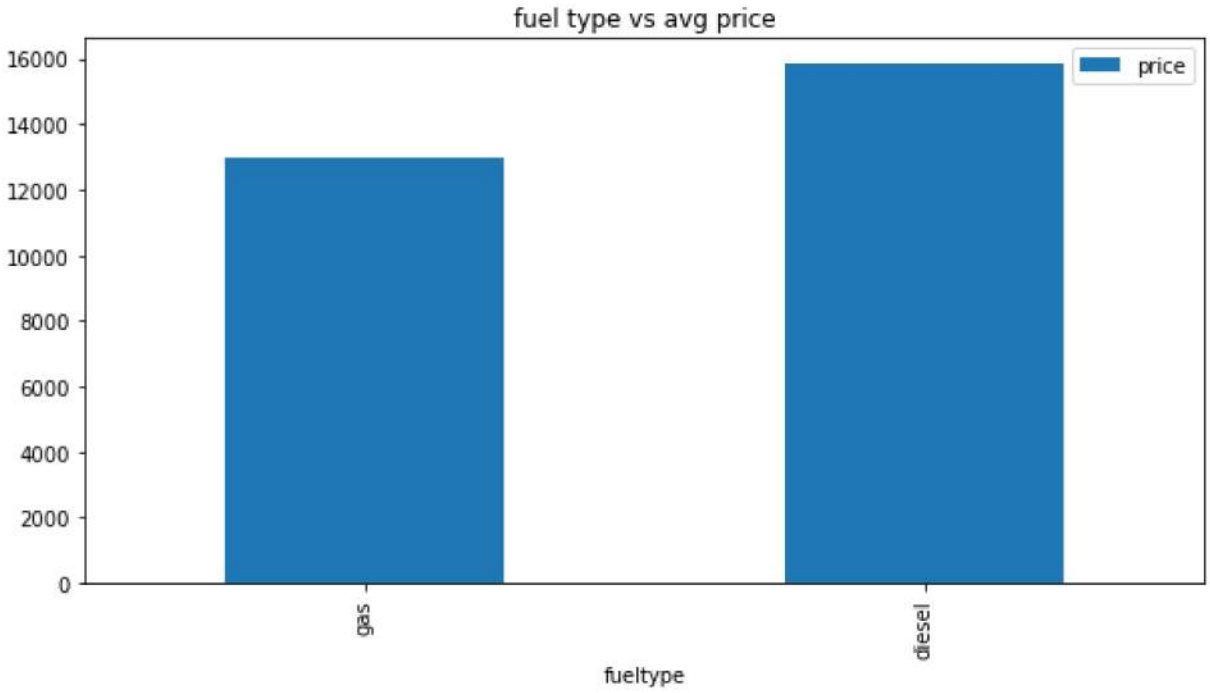
- ohc enginetype is used by most and dohcv is used by ver few.
- ohcv engine type has highest price, ohc and ohcf prices are almost similar.

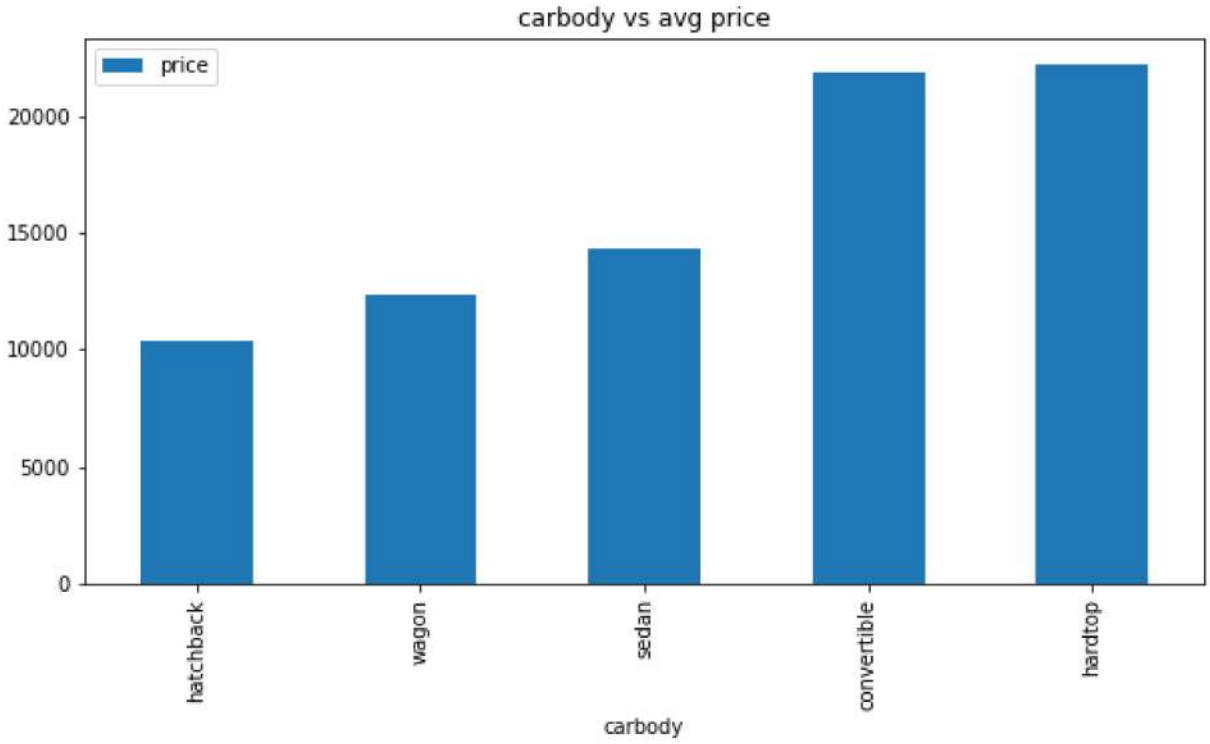
In [499…

```
plt7=pd.DataFrame(data.groupby(["CompanyName"])["price"].mean().sort_values())
plt7
plt.figure(figsize=(20,6))
#plt.subplot(1,3,1)
plt7.plot.bar(figsize=(10,5))
plt.title("company name vs avg price")
plt.show()

plt8=pd.DataFrame(data.groupby(["fueltype"])["price"].mean().sort_values())
plt8
plt8.plot.bar(figsize=(10,5))
plt.title("fuel type vs avg price")
plt.show()


plt9=pd.DataFrame(data.groupby(['carbody'])["price"].mean().sort_values())
plt9
plt9.plot.bar(figsize=(10,5))
plt.title("carbody vs avg price")
plt.show()
```

```
<Figure size 1440x432 with 0 Axes>
```







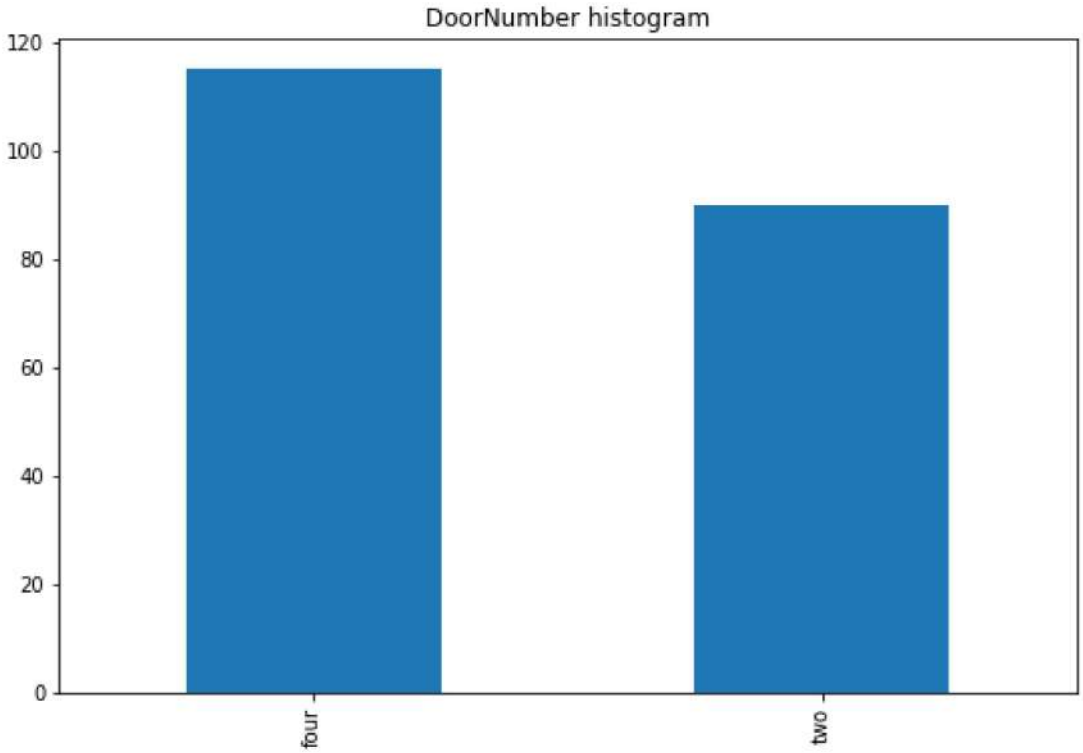
Inference for above

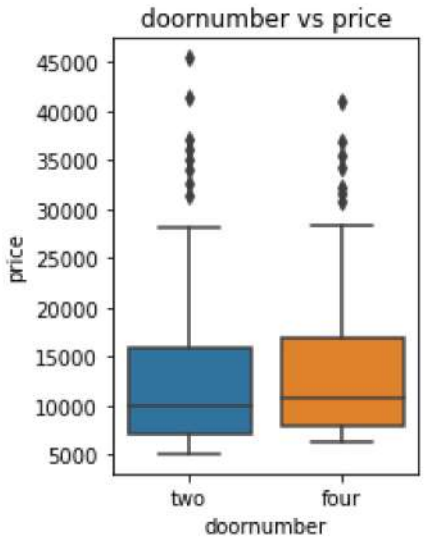
- buick and jaguar has high avg price
- diesel has avg price than gas
- convertible and hardtop has high avg price

In [500…

```
plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
plt.title("DoorNumber histogram")
plt9=data.doornumber.value_counts().plot(kind="bar")    #sns.countplot(data.doornumber)
#sns.displot(plt9)
```

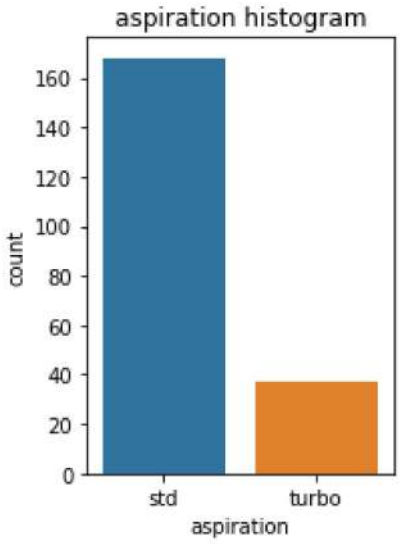
```
plt.show()
plt.subplot(1,2,2)
plt.title("doornumber vs price")
sns.boxplot(x=data.doornumber,y=data.price)
plt.show()

plt.subplot(1,2,1)
plt.title("aspiration histogram")
sns.countplot(data.aspiration)       # countplot is similar to value_counts function in
plt.show()

plt.subplot(1,2,2)
plt.title("aspiratioon vs price ")
sns.boxplot(x=data.aspiration,y=data.price)
plt.show()
```

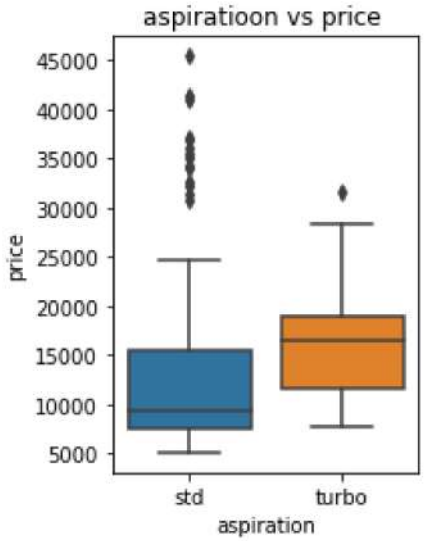





```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
```

```
result in an error or misinterpretation.
  warnings.warn(
```





Inference for above

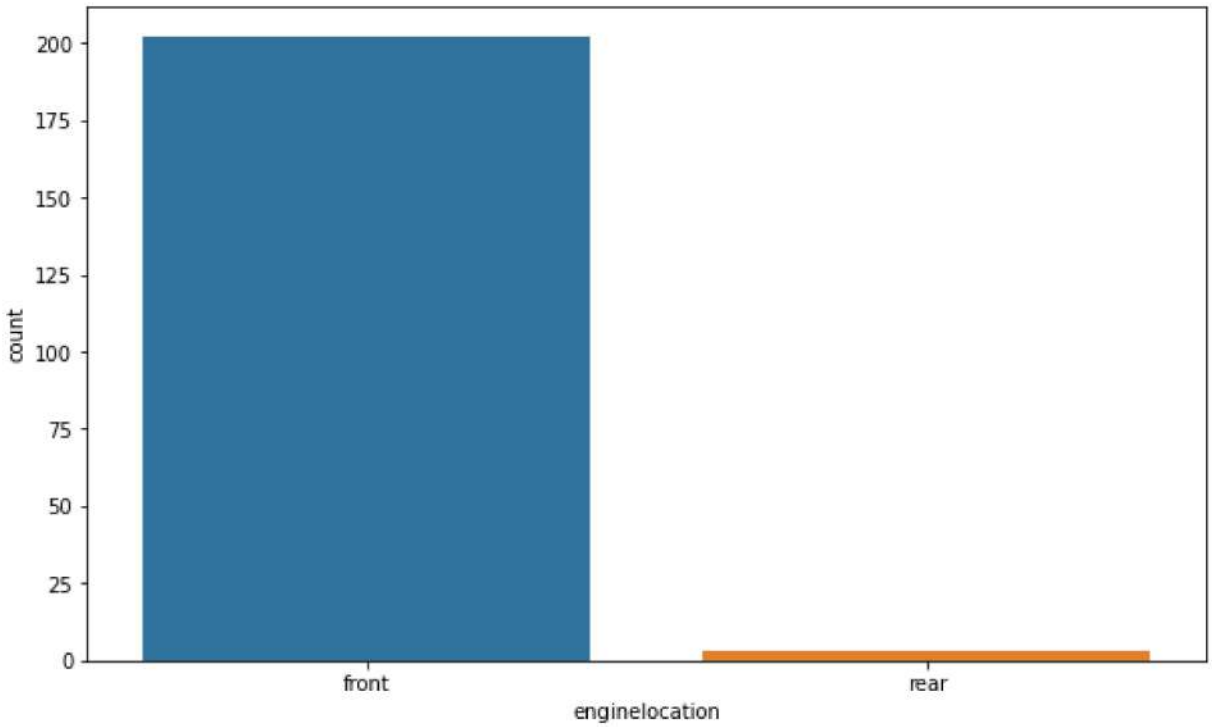
- door number variable is not effecting price much
- aspriration for turbo has some price range than std

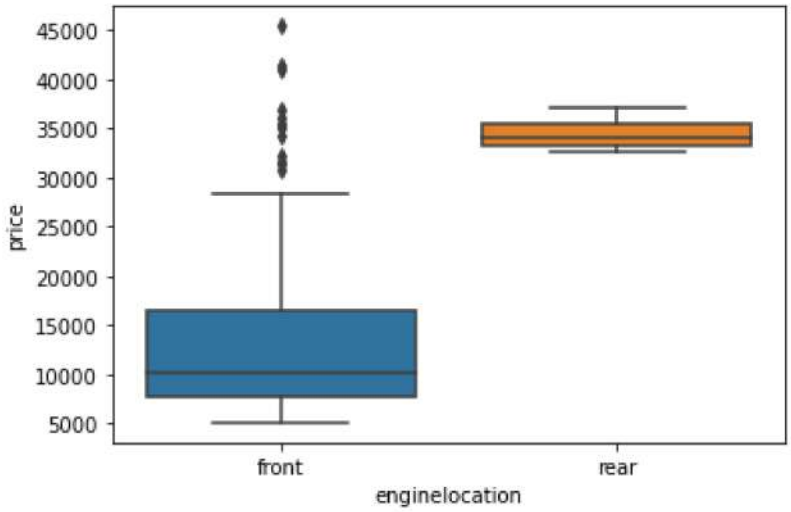
In [501…

```
plt.figure(figsize=(10,6))
sns.countplot(data.enginelocation)
plt.show()
sns.boxplot(x=data.enginelocation,y=data.price)
plt.show()


plt.figure(figsize=(10,6))
sns.countplot(data.cylindernumber)
plt.show()
sns.boxplot(x=data.cylindernumber,y=data.price)
plt.show()

plt.figure(figsize=(10,6))
sns.countplot(data.fuelsystem)
```

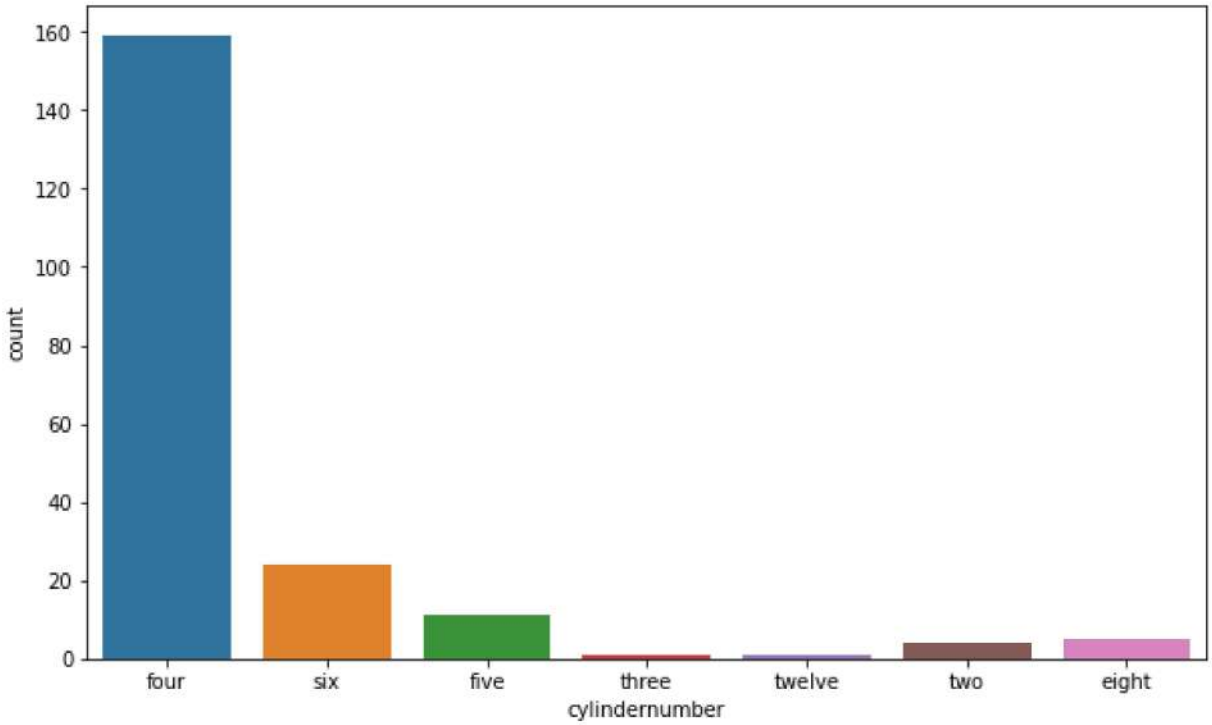
```
plt.show()
sns.boxplot(x=data.fuelsystem,y=data.price)
plt.show()

plt.figure(figsize=(10,6))
sns.countplot(data.drivewheel)
plt.show()
sns.boxplot(x=data.drivewheel,y=data.price)
plt.show()
```

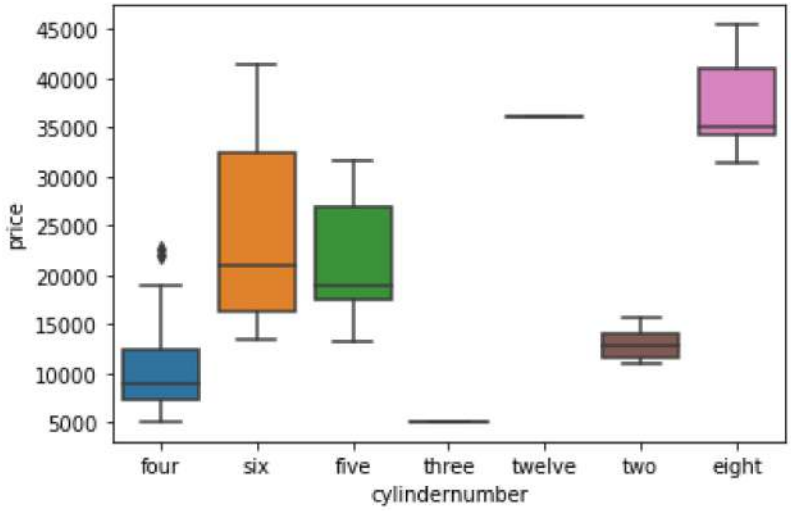
```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



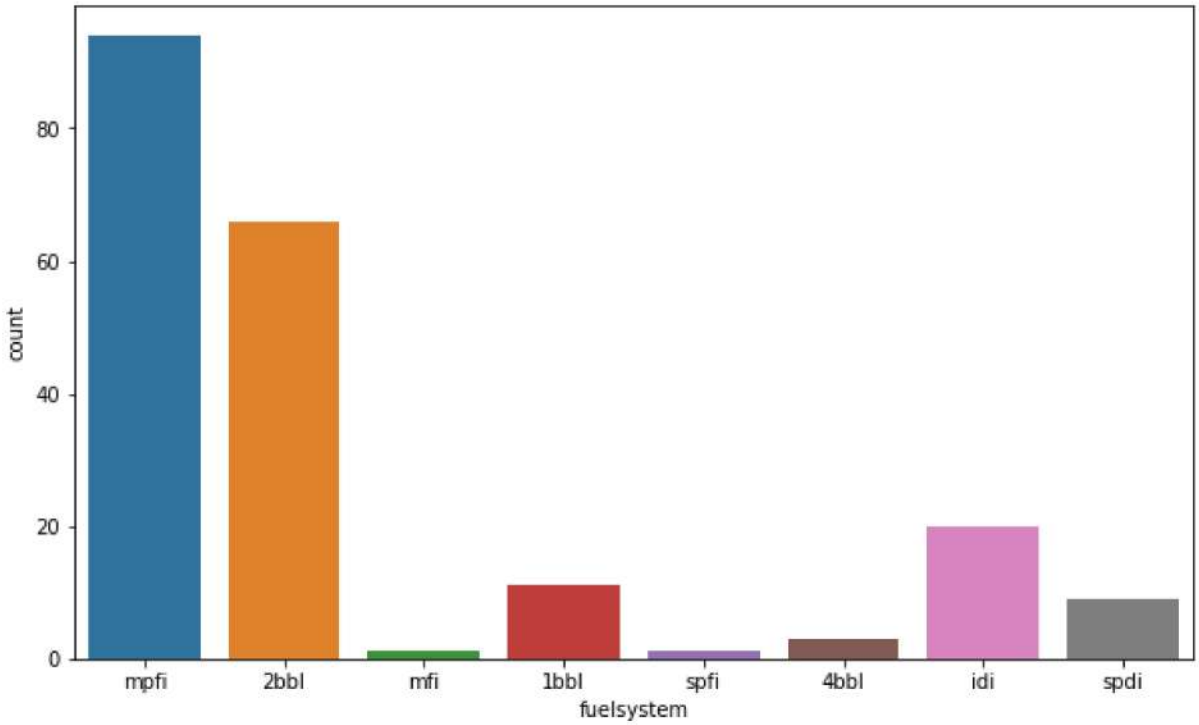


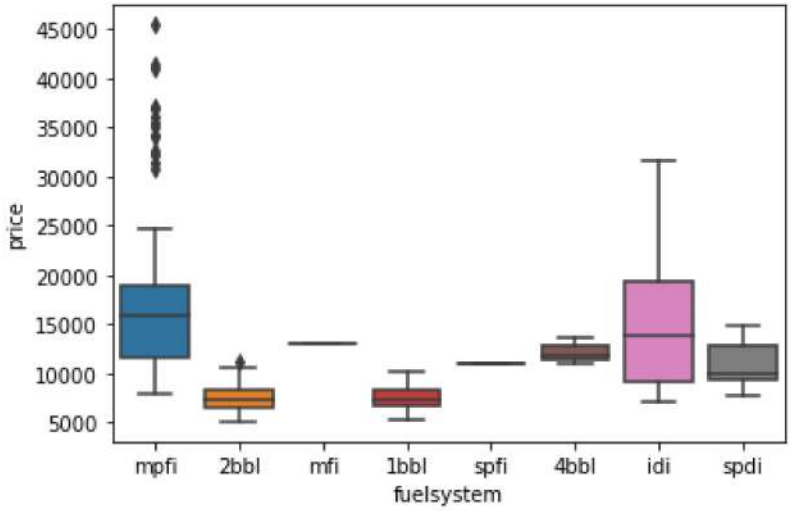

```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



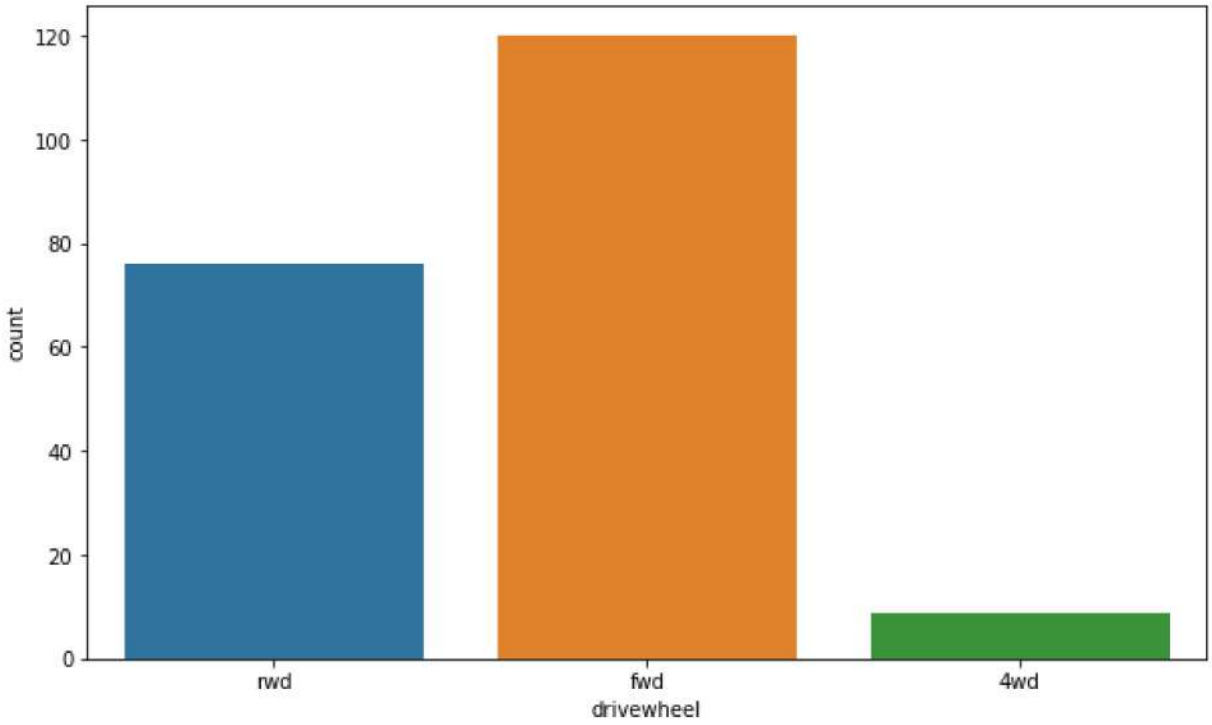


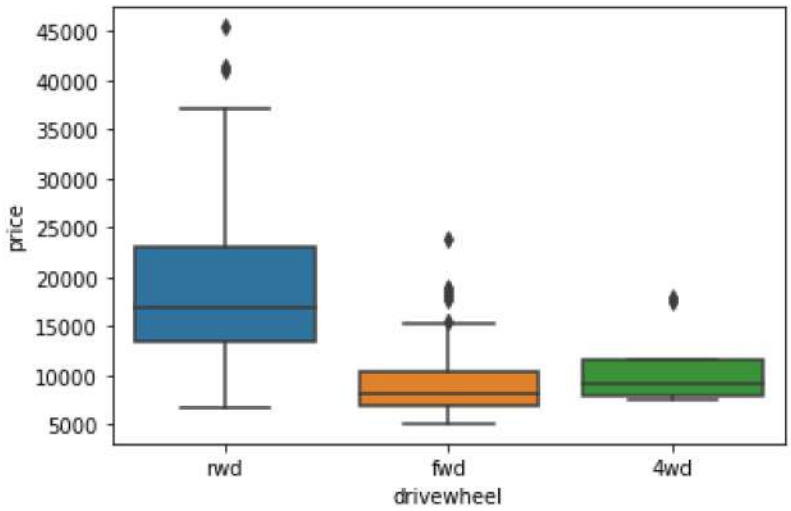

```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```






```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```





Inference for above

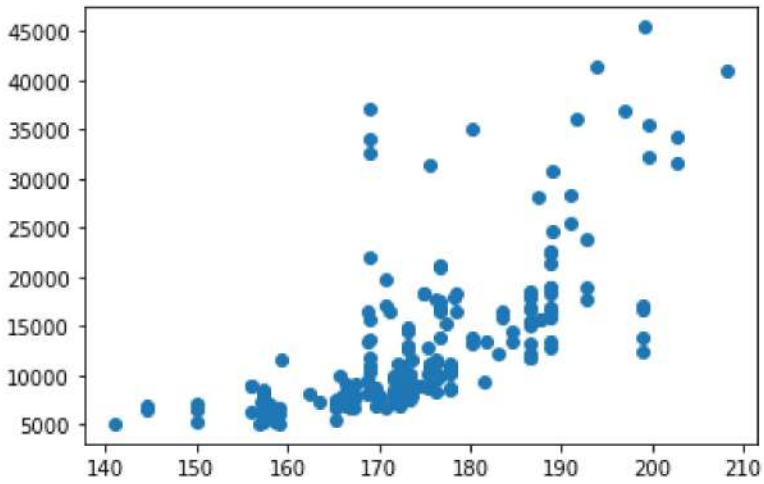
- enginelocation variable rear has high price still used by very few.
- cylinder with 4 5 6 are common but the price for 8 cylinders cars is high
- mpfi and 2bbl fuel system are most common type ...mpfi and idi having highest price range.
- fwd is preferred drive wheel but price is low for that..the price of rwd is ery high compafred to fewd and 4wd

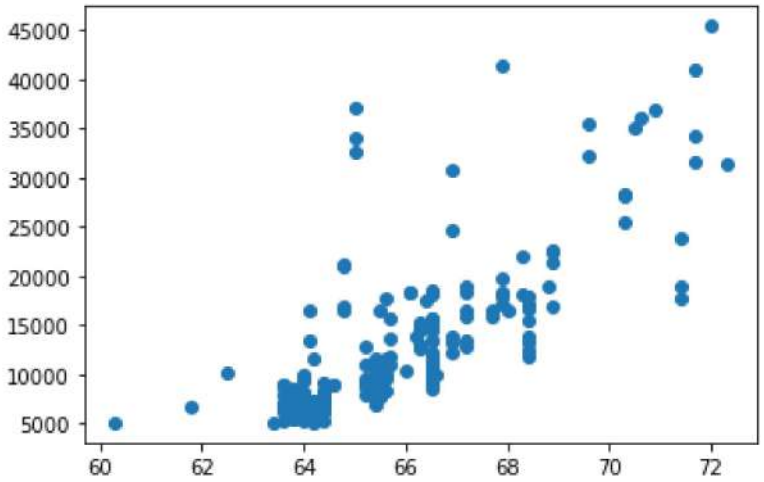
Visualizing Numerical Data

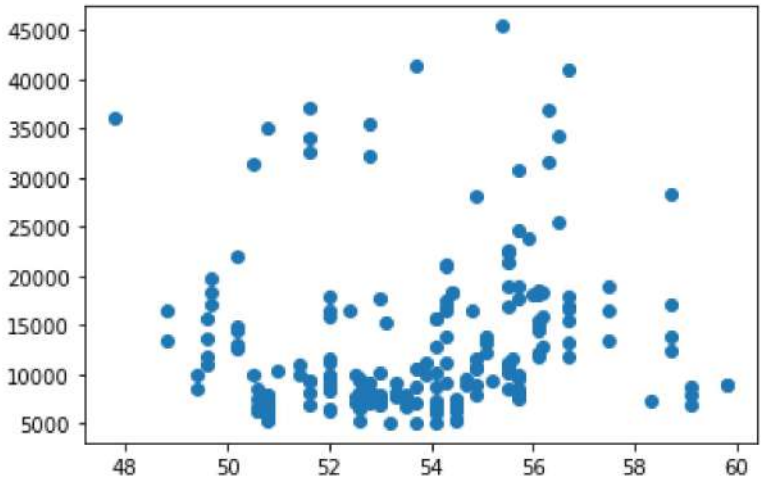
In [502…

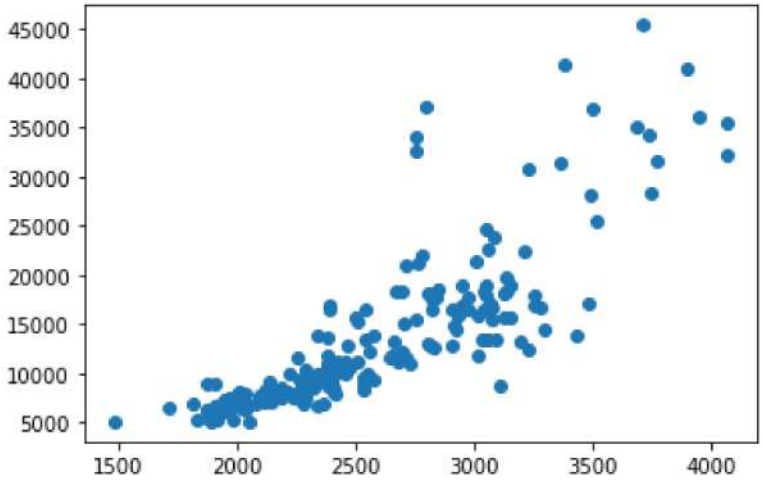
```
    # learn how to write code  to put xlabel ylabel by using a function
def plottingfn(i,j):
    #plt.xlabel()
    #plt.ylabel(j)
```

```
    plt.scatter(i,j)
    #plt.xlabel()
    plt.show()


plottingfn(data.carlength,data.price)
plottingfn(data.carwidth,data.price)
plottingfn(data.carheight,data.price)
plottingfn(data.curbweight,data.price)


#plt.figure(figsize=(10,6))
#plottingfn(data.carlength,data.price)
#plt.show()

#plt.scatter(x=data.carwidth,y=data.price)
#plt.show()

#plt.scatter(x=data.carheight,y=data.price)
#plt.show()

#plt.scatter(x=data.curbweight,y=data.price)
#plt.show()
```









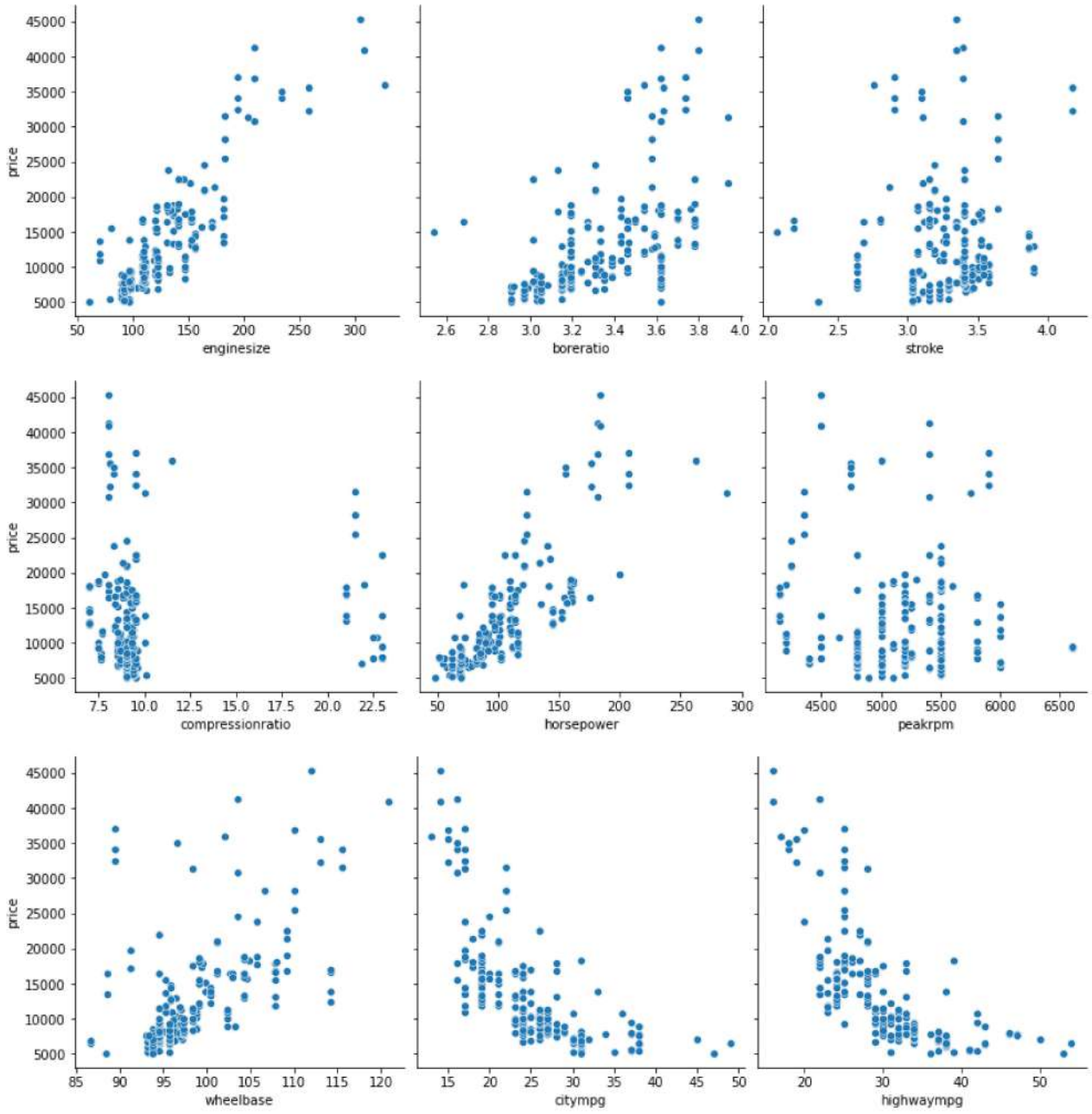
inference for above

- cralength carwidth curbweight show positive correlation with price
- car height does not have positive correlation with price data points are randomly distributed.

In [503…

```
import matplotlib
print(matplotlib.__version__)
def pp(x,y,z):
    sns.pairplot(data,x_vars=[x,y,z],y_vars="price",height=4,aspect=1,kind="scatter",di
                                                        # otheerwise first plot in belo
    plt.show()

pp("enginesize","boreratio","stroke")
pp("compressionratio","horsepower","peakrpm")
pp("wheelbase","citympg","highwaympg")
```

```
3.3.2
```



Inference for above

- enginesize, boreratio, horsepower, wheel base are in positive correlation with price
- citympg and highwaympg are negative correlation with price

In [504…
```
np.corrcoef(data.carlength,data.carwidth)   #correlation coefficient is used for derivi
                                            #the linear relationship b/w data points.
```

Out[504…
```
array([[1.        , 0.84111827],
       [0.84111827, 1.        ]])
```

Deriving New features

In [505…
```
# why we are not using for loop even thpugh it is printing for all the records in table
data["Fueleconomy"]=0.55*data["citympg"]+0.45*data["highwaympg"]
```

```
data["Fueleconomy"]
```

Out[505…
```
0      23.70
1      23.70
2      22.15
3      26.70
4      19.80
       ...
200    25.25
201    21.70
202    20.25
203    26.45
204    21.70
Name: Fueleconomy, Length: 205, dtype: float64
```

In [506…
```
#Binning the car companies based on average price of each company
data["price"]=data["price"].astype('int64')
data['price']  # data.price
```

Out[506…
```
0      13495
1      16500
2      16500
3      13950
4      17450
       ...
200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, Length: 205, dtype: int64
```

In [507…
```
temp=data.copy()
print(temp)
table=(temp.groupby(["CompanyName"]))["price"].mean()
table
```

```
     car_ID  symboling  CompanyName fueltype aspiration doornumber  \
0         1          3  alfa-romero      gas        std        two
1         2          3  alfa-romero      gas        std        two
2         3          1  alfa-romero      gas        std        two
3         4          2         audi      gas        std       four
4         5          2         audi      gas        std       four
..      ...        ...          ...      ...        ...        ...
200     201         -1        volvo      gas        std       four
201     202         -1        volvo      gas      turbo       four
202     203         -1        volvo      gas        std       four
203     204         -1        volvo   diesel      turbo       four
204     205         -1        volvo      gas      turbo       four

         carbody drivewheel enginelocation  wheelbase  ...  fuelsystem  \
0    convertible        rwd          front       88.6  ...        mpfi
1    convertible        rwd          front       88.6  ...        mpfi
2      hatchback        rwd          front       94.5  ...        mpfi
3          sedan        fwd          front       99.8  ...        mpfi
4          sedan        4wd          front       99.4  ...        mpfi
..           ...        ...            ...        ...  ...         ...
200        sedan        rwd          front      109.1  ...        mpfi
201        sedan        rwd          front      109.1  ...        mpfi
202        sedan        rwd          front      109.1  ...        mpfi
203        sedan        rwd          front      109.1  ...         idi
204        sedan        rwd          front      109.1  ...        mpfi

     boreratio  stroke  compressionratio horsepower peakrpm  citympg  \
```

```
0         3.47    2.68              9.0        111    5000        21
1         3.47    2.68              9.0        111    5000        21
2         2.68    3.47              9.0        154    5000        19
3         3.19    3.40             10.0        102    5500        24
4         3.19    3.40              8.0        115    5500        18
..         ...     ...              ...        ...     ...       ...
200       3.78    3.15              9.5        114    5400        23
201       3.78    3.15              8.7        160    5300        19
202       3.58    2.87              8.8        134    5500        18
203       3.01    3.40             23.0        106    4800        26
204       3.78    3.15              9.5        114    5400        19

    highwaympg  price  Fueleconomy
0           27  13495        23.70
1           27  16500        23.70
2           26  16500        22.15
3           30  13950        26.70
4           22  17450        19.80
..         ...    ...          ...
200         28  16845        25.25
201         25  19045        21.70
202         23  21485        20.25
203         27  22470        26.45
204         25  22625        21.70

[205 rows x 27 columns]
```

Out[507…

```
CompanyName
alfa-romero    15498.333333
audi           17859.142857
bmw            26118.750000
buick          33647.000000
chevrolet       6007.000000
dodge           7875.444444
honda           8184.692308
isuzu           8916.250000
jaguar         34600.000000
mazda          10652.882353
mercury        16503.000000
mitsubishi      9239.769231
nissan         10415.666667
peugeot        15489.090909
plymouth        7963.428571
porsche        31400.400000
renault         9595.000000
saab           15223.333333
subaru          8541.250000
toyota          9885.812500
volkswagen     10077.500000
volvo          18063.181818
Name: price, dtype: float64
```

In [508…

```
#for i in table:
 #   print(table[i])
""" if  table[i] > 0.0 and table[i] < 20000.0:
        data["carsrange"]='medium'
    elif table[i]>20000:
        data["carsrange"]='Highend'
data"""
```

Out[508…

```
' if  table[i] > 0.0 and table[i] < 20000.0:\n        data["carsrange"]=\'medium\'\n
elif table[i]>20000:\n        data["carsrange"]=\'Highend\'\ndata'
```

In [509…

```
print (table.reset_index())
temp=temp.merge(table.reset_index(),how='left',on='CompanyName')   #left outer join  us
```

```
# on may be column or index level names to join on..   must be present in both datafram
# suffoxes can be manually given not only _x and _y  we can give whatever names we want

temp
```

```
    CompanyName         price
0   alfa-romero  15498.333333
1          audi  17859.142857
2           bmw  26118.750000
3         buick  33647.000000
4     chevrolet   6007.000000
5         dodge   7875.444444
6         honda   8184.692308
7         isuzu   8916.250000
8        jaguar  34600.000000
9         mazda  10652.882353
10      mercury  16503.000000
11   mitsubishi   9239.769231
12       nissan  10415.666667
13      peugeot  15489.090909
14     plymouth   7963.428571
15      porsche  31400.400000
16      renault   9595.000000
17         saab  15223.333333
18       subaru   8541.250000
19       toyota   9885.812500
20   volkswagen  10077.500000
21        volvo  18063.181818
```

Out[509…

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	eng
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	
...	...	...	...	...	...	...	...	...	...
200	201	-1	volvo	gas	std	four	sedan	rwd	
201	202	-1	volvo	gas	turbo	four	sedan	rwd	
202	203	-1	volvo	gas	std	four	sedan	rwd	
203	204	-1	volvo	diesel	turbo	four	sedan	rwd	
204	205	-1	volvo	gas	turbo	four	sedan	rwd	

205 rows × 28 columns

In [510…

```
bins=[0,10000,20000,40000]     # 0 to 10000 and 10000 to 20000 and 20000 to 40000
cars_bin=["budget","medium","highend"]  #bin labels must be one smaller than no.of bins
data["Carsrange"]=pd.cut(temp['price_y'],bins,labels=cars_bin) # bin values into discre
# cut function is useful for converting continuos variable to a categorical variable.
data
```

Out[510…

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	eng

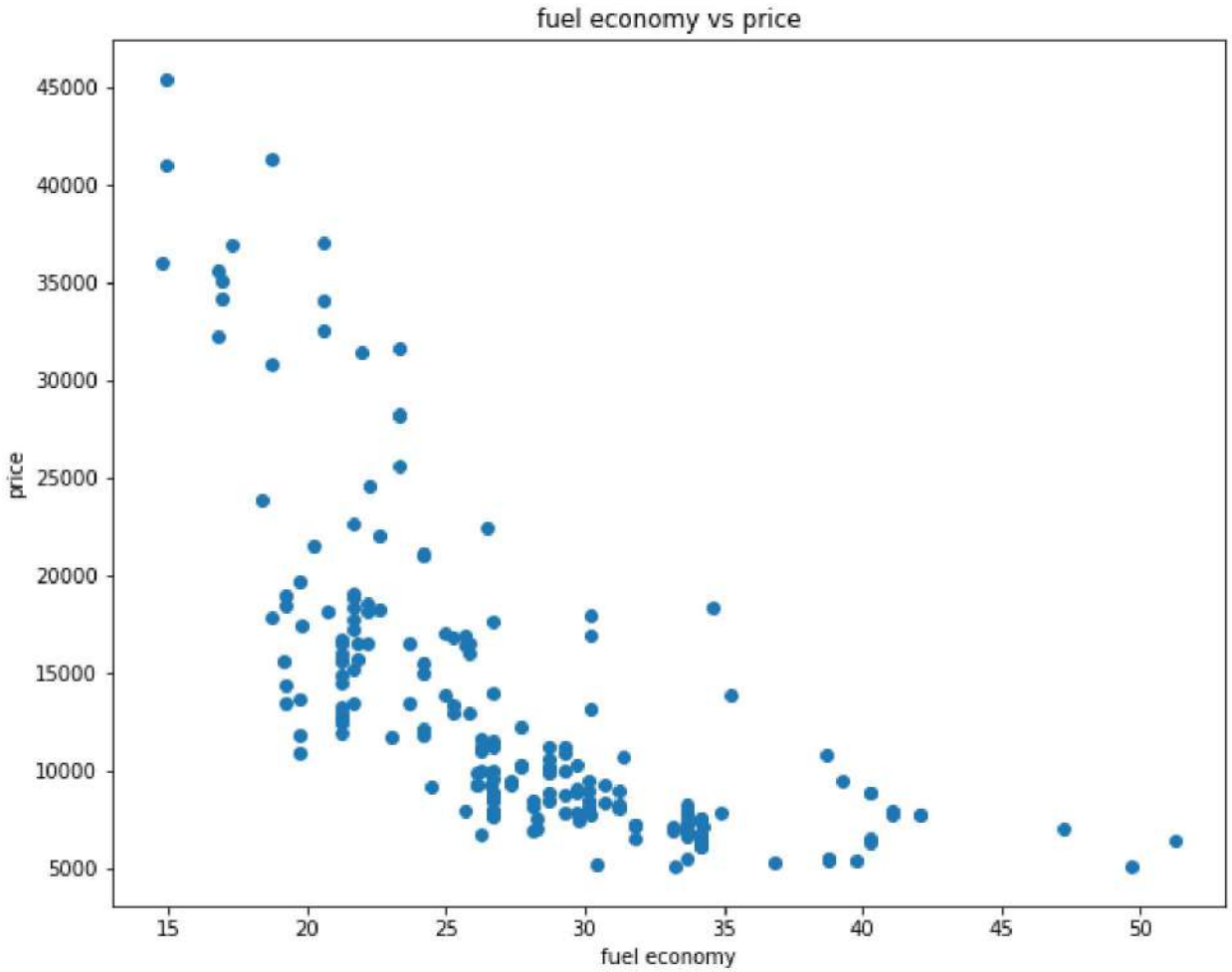
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	eng
0	1	3	alfa-romero	gas	std	two	convertible	rwd	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	
3	4	2	audi	gas	std	four	sedan	fwd	
4	5	2	audi	gas	std	four	sedan	4wd	
...	...	...	...	...	...	...	...	...	
200	201	-1	volvo	gas	std	four	sedan	rwd	
201	202	-1	volvo	gas	turbo	four	sedan	rwd	
202	203	-1	volvo	gas	std	four	sedan	rwd	
203	204	-1	volvo	diesel	turbo	four	sedan	rwd	
204	205	-1	volvo	gas	turbo	four	sedan	rwd	

205 rows × 28 columns

Bivariate Analysis

In [511…

```
plt.figure(figsize=(10,8))
plt.title("fuel economy vs price")
plt.scatter(data['Fueleconomy'],data.price)

plt.xlabel("fuel economy")
plt.ylabel("price")
plt.show()
```



Inference for above

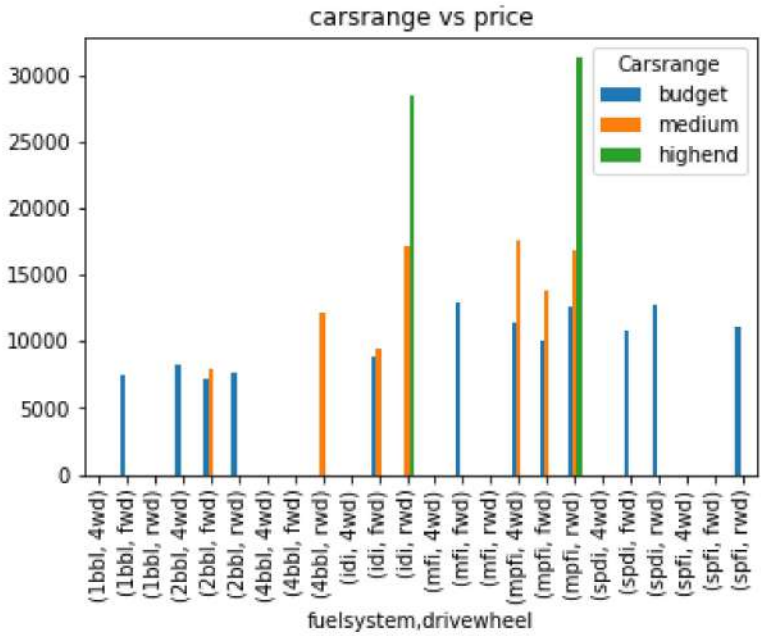
Fuel economy and price are negatively correlated.

In [512…

```
plt.figure(figsize=(10,8))

z=pd.DataFrame(data.groupby(["fuelsystem","drivewheel","Carsrange"])["price"].mean().un
print(z)

z.plot.bar()
plt.title("carsrange vs price")
plt.show()

#data.groupby(["Carsrange"])["price"].mean()
```

Carsrange		budget	medium	highend
fuelsystem	drivewheel			
1bbl	4wd	NaN	NaN	NaN
	fwd	7555.545455	NaN	NaN
	rwd	NaN	NaN	NaN
2bbl	4wd	8305.000000	NaN	NaN
	fwd	7126.000000	7870.904762	NaN
	rwd	7693.666667	NaN	NaN
4bbl	4wd	NaN	NaN	NaN
	fwd	NaN	NaN	NaN
	rwd	NaN	12145.000000	NaN
idi	4wd	NaN	NaN	NaN

```
           fwd          8794.666667   9500.666667       NaN
           rwd                  NaN  17114.142857  28394.0
mfi        4wd                  NaN           NaN       NaN
           fwd         12964.000000           NaN       NaN
           rwd                  NaN           NaN       NaN
mpfi       4wd         11476.500000  17654.500000       NaN
           fwd          9990.000000  13830.090909       NaN
           rwd         12610.500000  16793.600000  31267.6
spdi       4wd                  NaN           NaN       NaN
           fwd         10768.750000           NaN       NaN
           rwd         12764.000000           NaN       NaN
spfi       4wd                  NaN           NaN       NaN
           fwd                  NaN           NaN       NaN
           rwd         11048.000000           NaN       NaN
<Figure size 720x576 with 0 Axes>
```



Inference for above

highend cars uses rwd drive wheel and (idi and mpfi) as fuel system

List of significant variabes after visula analysis

- engine size
- boreratio
- horsepower
- wheelbase
- engine type -- categorical
- car body -- categorical
- fuel type -- categorical
- aspiration -- categorical
- #engine location 😊
- cylinder number -- categorical
- #fuel system 😊

- drive wheel -- categorical
- fuel economy
- car range -- categorical
- car length
- carwidth
- curbweight
- #citympg 😊
- #highwaympg 😊 the above two are negatively correlated with price and did not consider.

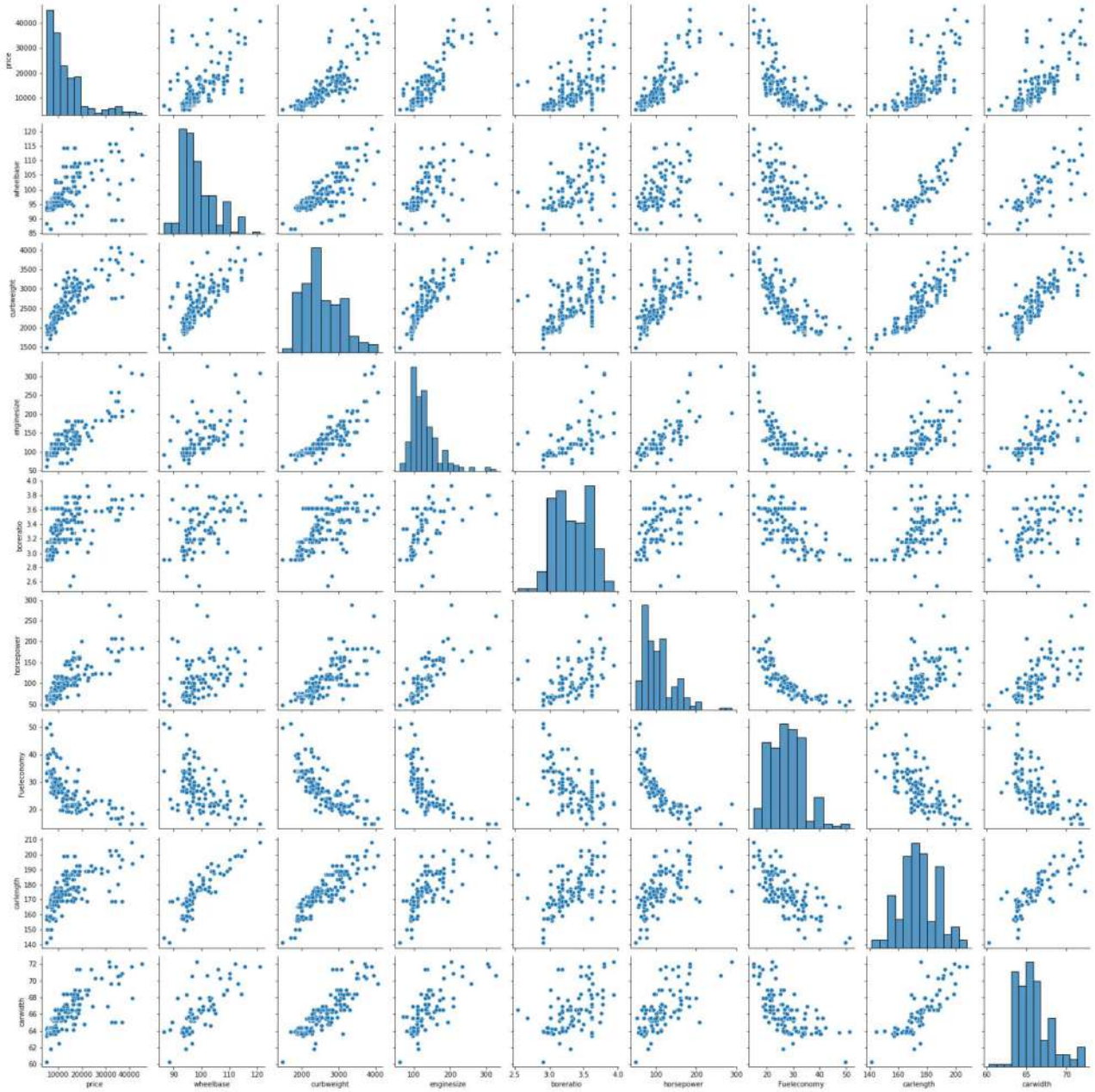
In [513…
```
data_lr=data[['price','fueltype','aspiration','carbody','drivewheel','wheelbase','curbw
              'enginesize','boreratio','horsepower','Fueleconomy','carlength','carwidth
data_lr
```

Out[513…

	price	fueltype	aspiration	carbody	drivewheel	wheelbase	curbweight	enginetype	cylindern
0	13495	gas	std	convertible	rwd	88.6	2548	dohc	
1	16500	gas	std	convertible	rwd	88.6	2548	dohc	
2	16500	gas	std	hatchback	rwd	94.5	2823	ohcv	
3	13950	gas	std	sedan	fwd	99.8	2337	ohc	
4	17450	gas	std	sedan	4wd	99.4	2824	ohc	
...	...	...	...	...	...	...	...	...	
200	16845	gas	std	sedan	rwd	109.1	2952	ohc	
201	19045	gas	turbo	sedan	rwd	109.1	3049	ohc	
202	21485	gas	std	sedan	rwd	109.1	3012	ohcv	
203	22470	diesel	turbo	sedan	rwd	109.1	3217	ohc	
204	22625	gas	turbo	sedan	rwd	109.1	3062	ohc	

205 rows × 16 columns

In [514…
```
sns.pairplot(data_lr)   # seaborn will show only numeric values
plt.show()
```



Dummy variables

- for categorical variables, we need to create dummy variables

In [515…

```
def dummies(x,y):
    temp=pd.get_dummies(y[x],drop_first=True)
    y=pd.concat([y,temp],axis=1)
    y.drop([x],axis=1,inplace=True)
    return y



"""temp=pd.get_dummies(data_lr['fueltype'],drop_first=True) #drop_first bool, default F
# to get k-1 dummies out of k categorical levels by removing the first level
temp   #if drop_firts is TRue , first column is dropped.  for k variables, we get k-1 d
df=pd.concat([data_lr,temp],axis=1) # axis =1 means concatenate along columns, and 0 me
print(df)
df.drop('fueltype',axis=1,inplace=True) #  if axis is not menetioned i.e default =0(alo
```

```
#then it will raise error because fueltypeis not found in rows and inplace must be TRue
df"""


data_lr=dummies('fueltype',data_lr)
data_lr=dummies('aspiration',data_lr)
data_lr=dummies('carbody',data_lr)
data_lr=dummies('drivewheel',data_lr)
data_lr=dummies('enginetype',data_lr)
data_lr=dummies("cylindernumber",data_lr)
data_lr=dummies("Carsrange",data_lr)

data_lr
```

Out[515…

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	Fueleconomy	carlength	carwi
0	13495	88.6	2548	130	3.47	111	23.70	168.8	€
1	16500	88.6	2548	130	3.47	111	23.70	168.8	€
2	16500	94.5	2823	152	2.68	154	22.15	171.2	€
3	13950	99.8	2337	109	3.19	102	26.70	176.6	€
4	17450	99.4	2824	136	3.19	115	19.80	176.6	€
...	...	...	...	...	...	...	...	...	
200	16845	109.1	2952	141	3.78	114	25.25	188.8	€
201	19045	109.1	3049	141	3.78	160	21.70	188.8	€
202	21485	109.1	3012	173	3.58	134	20.25	188.8	€
203	22470	109.1	3217	145	3.01	106	26.45	188.8	€
204	22625	109.1	3062	141	3.78	114	21.70	188.8	€

205 rows × 31 columns

Train Test split and scaling

In [516…

```
from sklearn.model_selection import train_test_split
np.random.seed(0)   # same output for each time.
df_train,df_test=train_test_split(data_lr,train_size=0.7,test_size=0.3,random_state=100
#then everytime(for each execution) different values for train and test sets
# if random_state = any integer  then the the order/list of values in test and train se
# if random_state=0  one order is fixed even if we execute multiple times
#  if random_state=1  different order from above will be produced
# if random state= 2,3,4,.....different orders for each integers and the order is fixe
print(df_train,df_test)
```

```
     price  wheelbase  curbweight  enginesize  boreratio  horsepower  \
122   7609       93.7        2191          98       2.97          68
125  22018       94.5        2778         151       3.94         143
166   9538       94.5        2300          98       3.24         112
1    16500       88.6        2548         130       3.47         111
199  18950      104.3        3157         130       3.62         162
..     ...        ...         ...         ...        ...         ...
87    9279       96.3        2403         110       3.17         116
```

```
103  13499      100.4        3060         181       3.43         152
67   25552      110.0        3515         183       3.58         123
24    6229       93.7        1967          90       2.97          68
8    23875      105.8        3086         131       3.13         140

     Fueleconomy  carlength  carwidth  gas  ...  ohcv  rotor  five  four  six  \
122        34.15      167.3      63.8    1  ...     0      0     0     1    0
125        22.60      168.9      68.3    1  ...     0      0     0     1    0
166        27.35      168.7      64.0    1  ...     0      0     0     1    0
1          23.70      168.8      64.1    1  ...     0      0     0     1    0
199        19.25      188.8      67.2    1  ...     0      0     0     1    0
..           ...        ...       ...  ...  ...   ...    ...   ...   ...  ...
87         26.15      172.4      65.4    1  ...     0      0     0     1    0
103        21.70      184.6      66.5    1  ...     1      0     0     0    1
67         23.35      190.9      70.3    0  ...     0      0     1     0    0
24         34.15      157.3      63.8    1  ...     0      0     0     1    0
8          18.35      192.7      71.4    1  ...     0      0     1     0    0

     three  twelve  two  medium  highend
122      0       0    0       0        0
125      0       0    0       0        1
166      0       0    0       0        0
1        0       0    0       1        0
199      0       0    0       1        0
..     ...     ...  ...     ...      ...
87       0       0    0       0        0
103      0       0    0       1        0
67       0       0    0       0        1
24       0       0    0       0        0
8        0       0    0       1        0

[143 rows x 31 columns]      price  wheelbase  curbweight  enginesize  boreratio  horsep
ower  \
160   7738       95.7        2094          98       3.19          70
186   8495       97.3        2275         109       3.19          85
59    8845       98.8        2385         122       3.39          84
165   9298       94.5        2265          98       3.24         112
140   7603       93.3        2240         108       3.62          73
..     ...        ...         ...         ...        ...         ...
28    8921      103.3        2535         122       3.34          88
29   12964       95.9        2811         156       3.60         145
182   7775       97.3        2261          97       3.01          52
40   10295       96.5        2372         110       3.15          86
128  37028       89.5        2800         194       3.74         207

     Fueleconomy  carlength  carwidth  gas  ...  ohcv  rotor  five  four  six  \
160        42.05      166.3      64.4    1  ...     0      0     0     1    0
186        30.15      171.7      65.5    1  ...     0      0     0     1    0
59         28.70      177.8      66.5    1  ...     0      0     0     1    0
165        27.35      168.7      64.0    1  ...     0      0     0     1    0
140        28.25      157.3      63.8    1  ...     0      0     0     1    0
..           ...        ...       ...  ...  ...   ...    ...   ...   ...  ...
28         26.70      174.6      64.6    1  ...     0      0     0     1    0
29         21.25      173.2      66.3    1  ...     0      0     0     1    0
182        41.05      171.7      65.5    0  ...     0      0     0     1    0
40         29.70      175.4      62.5    1  ...     0      0     0     1    0
128        20.60      168.9      65.0    1  ...     0      0     0     0    1

     three  twelve  two  medium  highend
160      0       0    0       0        0
186      0       0    0       1        0
59       0       0    0       1        0
165      0       0    0       0        0
140      0       0    0       0        0
..     ...     ...  ...     ...      ...
```

```
28      0       0    0       0       0
29      0       0    0       0       0
182     0       0    0       1       0
40      0       0    0       0       0
128     0       0    0       0       1

[62 rows x 31 columns]
```

In [517…

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
print(scaler)
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepower','Fueleco
df_train[num_vars]=scaler.fit_transform(df_train[num_vars])
df_train
```

```
MinMaxScaler()
<ipython-input-517-43284997117e>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df_train[num_vars]=scaler.fit_transform(df_train[num_vars])
C:\Users\prasa\anaconda3\lib\site-packages\pandas\core\indexing.py:1736: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
```

Out[517…

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	Fueleconomy	carlength	car
122	0.068818	0.244828	0.272692	0.139623	0.230159	0.083333	0.530864	0.426016	0.2
125	0.466890	0.272414	0.500388	0.339623	1.000000	0.395833	0.213992	0.452033	0.6
166	0.122110	0.272414	0.314973	0.139623	0.444444	0.266667	0.344307	0.448780	0.3
1	0.314446	0.068966	0.411171	0.260377	0.626984	0.262500	0.244170	0.450407	0.3
199	0.382131	0.610345	0.647401	0.260377	0.746032	0.475000	0.122085	0.775610	0.5
…	…	…	…	…	…	…	…	…	
87	0.114954	0.334483	0.354926	0.184906	0.388889	0.283333	0.311385	0.508943	0.4
103	0.231539	0.475862	0.609775	0.452830	0.595238	0.433333	0.189300	0.707317	0.5
67	0.564522	0.806897	0.786268	0.460377	0.714286	0.312500	0.234568	0.809756	0.8
24	0.030693	0.244828	0.185803	0.109434	0.230159	0.083333	0.530864	0.263415	0.2
8	0.518192	0.662069	0.619860	0.264151	0.357143	0.383333	0.097394	0.839024	0.9

143 rows × 31 columns

In [518…

```
df_train.describe()  # we can use percentiles and include parameter for all rows and fe
```

Out[518…

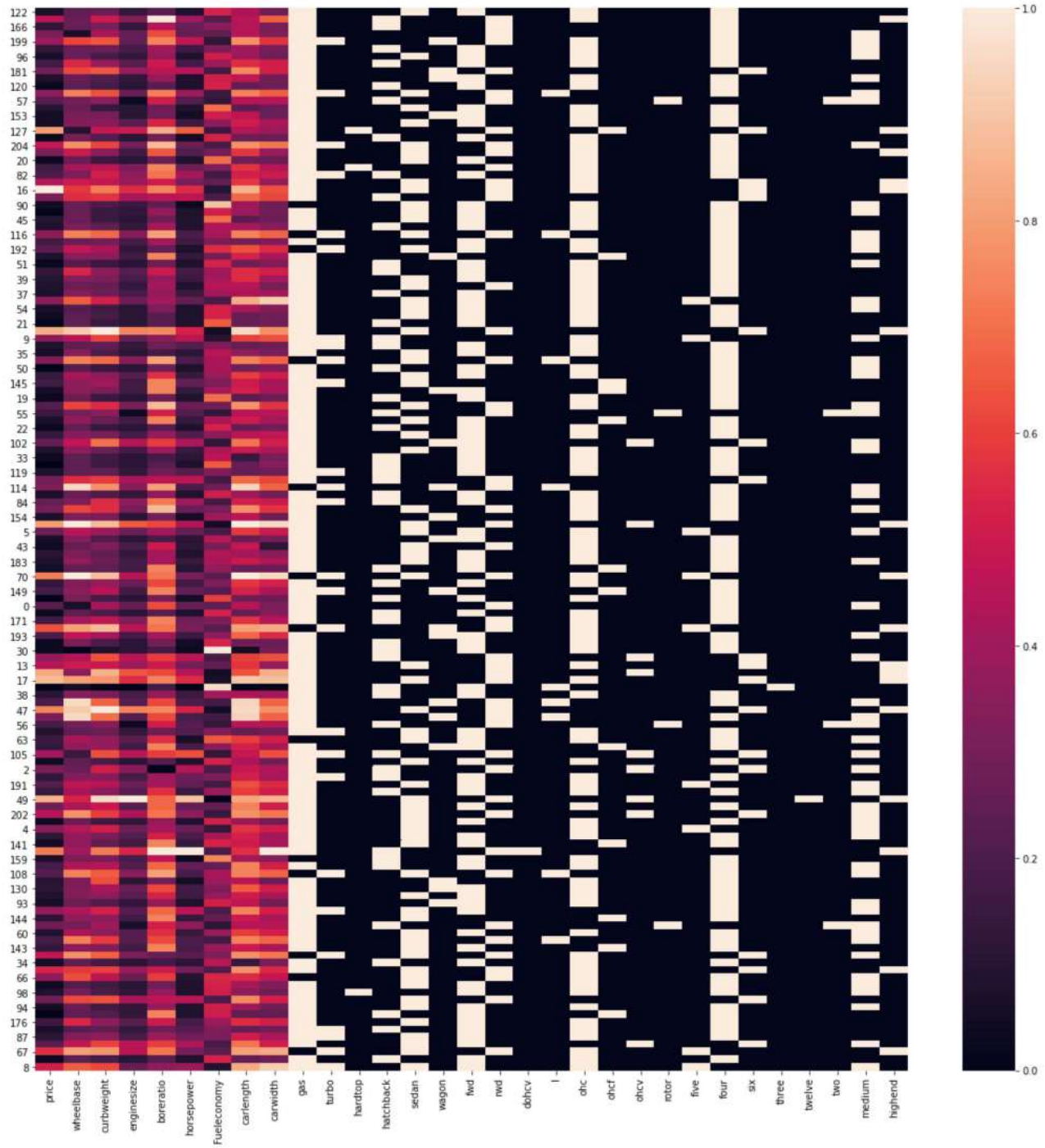
	price	wheelbase	curbweight	enginesize	boreratio	horsepower	Fueleconomy	carleng
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.0000

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	Fueleconomy	carleng
mean	0.219309	0.411141	0.407878	0.241351	0.497946	0.227302	0.358265	0.5254
std	0.215682	0.205581	0.211269	0.154619	0.207140	0.165511	0.185980	0.2048
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.067298	0.272414	0.245539	0.135849	0.305556	0.091667	0.198903	0.3991
50%	0.140343	0.341379	0.355702	0.184906	0.500000	0.191667	0.344307	0.5024
75%	0.313479	0.503448	0.559542	0.301887	0.682540	0.283333	0.512346	0.6699
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

8 rows × 31 columns

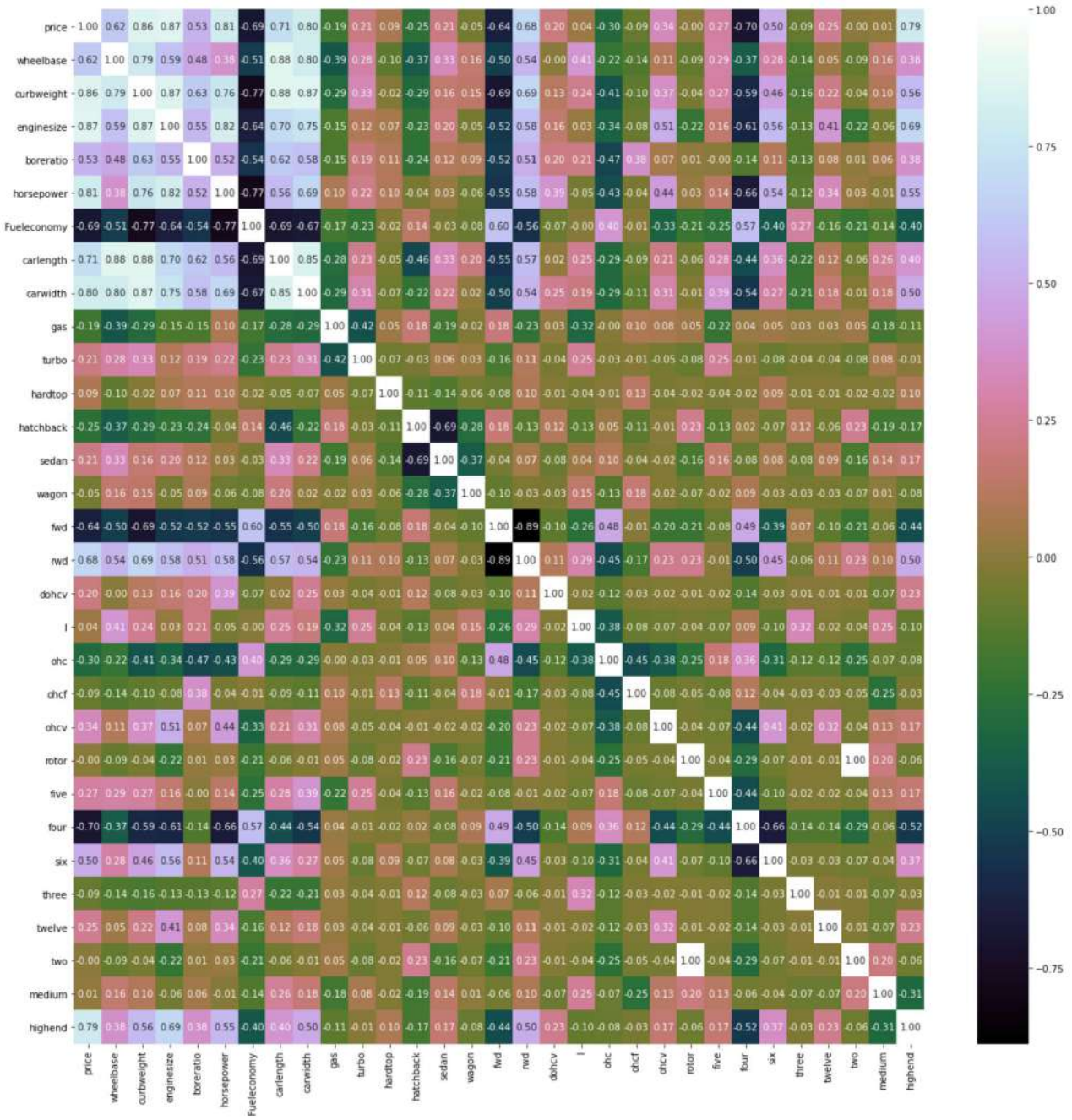
In [519…
```
# correlation using heatmap
plt.figure(figsize=(20,20))
sns.heatmap(df_train)
```

Out[519… <AxesSubplot:>



In [520…
```
plt.figure(figsize=(20,20))
sns.heatmap(df_train.corr(),annot=True,cmap='cubehelix',fmt='0.2f')  # annot=true fills
#fmt=0.2 means upto 2 decimal points
```

Out[520… <AxesSubplot:>



In [521…

```
# divide the data into x and y labels
y_train=df_train.pop("price")
X_train=df_train
print(y_train)  # if we run the cell again , we will get key error saying price is not
#so second time when we run, there is no price attribute in df_train..... for that we n
#x_train
```

```
122    0.068818
125    0.466890
166    0.122110
1      0.314446
199    0.382131
         ...
87     0.114954
103    0.231539
67     0.564522
24     0.030693
8      0.518192
Name: price, Length: 143, dtype: float64
```

In [522…
```
X_train
```

Out[522…

	wheelbase	curbweight	enginesize	boreratio	horsepower	Fueleconomy	carlength	carwidth	ga
122	0.244828	0.272692	0.139623	0.230159	0.083333	0.530864	0.426016	0.291667	
125	0.272414	0.500388	0.339623	1.000000	0.395833	0.213992	0.452033	0.666667	
166	0.272414	0.314973	0.139623	0.444444	0.266667	0.344307	0.448780	0.308333	
1	0.068966	0.411171	0.260377	0.626984	0.262500	0.244170	0.450407	0.316667	
199	0.610345	0.647401	0.260377	0.746032	0.475000	0.122085	0.775610	0.575000	
...	...	...	...	...	...	...	...	...	
87	0.334483	0.354926	0.184906	0.388889	0.283333	0.311385	0.508943	0.425000	
103	0.475862	0.609775	0.452830	0.595238	0.433333	0.189300	0.707317	0.516667	
67	0.806897	0.786268	0.460377	0.714286	0.312500	0.234568	0.809756	0.833333	
24	0.244828	0.185803	0.109434	0.230159	0.083333	0.530864	0.263415	0.291667	
8	0.662069	0.619860	0.264151	0.357143	0.383333	0.097394	0.839024	0.925000	

143 rows × 30 columns

Model Building

In [523…
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
lm=LinearRegression()
a=lm.fit(X_train,y_train)
print(a)
print(a.coef_) # weight vector of shape(no.of targets, no.of features)=(1,30)   e here
print(a.intercept_)  #-0.006477981385304865   is bias
rfe = RFE(a, n_features_to_select=10)
print(rfe)
rfe.fit(X_train,y_train)
print(rfe)
print(rfe.support_)  # returns True for all selected variables and fase for not slected
rfe.ranking_  # returns rank 1 for all features which are true and

# I did not understand what fetaures it selected
```

```
LinearRegression()
[ 1.09917927e-01  2.85740288e-01  4.51228270e-02 -1.05074895e-01
  6.06995546e-01  1.33917883e-01 -5.67971232e-02  2.08057088e-01
 -2.03824004e-02 -9.71999748e-03 -9.93611025e-02 -1.38807926e-01
 -1.20879861e-01 -1.40201624e-01 -1.80914517e-02  1.49433054e-02
 -3.30602132e-01 -2.94962337e-03  7.55115587e-02  1.16000104e-01
 -4.26281995e-02 -2.13041170e-04 -1.23643802e-01 -1.04593989e-01
 -8.27321584e-02  7.17557484e-02 -1.75216607e-01 -2.13041170e-04
  4.54977781e-02  2.26058650e-01]
-0.006477981385304865
RFE(estimator=LinearRegression(), n_features_to_select=10)
RFE(estimator=LinearRegression(), n_features_to_select=10)
```

```
[False  True False False  True  True False  True False False False  True
  True  True False False  True False False False False False False False
 False False  True False False  True]
```

Out[523…
```
array([ 3,  1, 13, 10,  1,  1, 11,  1, 17, 18,  2,  1,  1,  1, 16, 15,  1,
       19,  7,  8,  9, 21,  6,  4,  5, 14,  1, 20, 12,  1])
```

In [524…
```
a=list(zip(X_train.columns, rfe.support_,rfe.ranking_))
(a)
```

Out[524…
```
[('wheelbase', False, 3),
 ('curbweight', True, 1),
 ('enginesize', False, 13),
 ('boreratio', False, 10),
 ('horsepower', True, 1),
 ('Fueleconomy', True, 1),
 ('carlength', False, 11),
 ('carwidth', True, 1),
 ('gas', False, 17),
 ('turbo', False, 18),
 ('hardtop', False, 2),
 ('hatchback', True, 1),
 ('sedan', True, 1),
 ('wagon', True, 1),
 ('fwd', False, 16),
 ('rwd', False, 15),
 ('dohcv', True, 1),
 ('l', False, 19),
 ('ohc', False, 7),
 ('ohcf', False, 8),
 ('ohcv', False, 9),
 ('rotor', False, 21),
 ('five', False, 6),
 ('four', False, 4),
 ('six', False, 5),
 ('three', False, 14),
 ('twelve', True, 1),
 ('two', False, 20),
 ('medium', False, 12),
 ('highend', True, 1)]
```

In [525…
```
print(X_train.columns[rfe.ranking_])
X_train_rfe=X_train[X_train.columns[rfe.support_]]
X_train_rfe
```

```
Index(['boreratio', 'curbweight', 'wagon', 'hardtop', 'curbweight',
       'curbweight', 'hatchback', 'curbweight', 'l', 'ohc', 'enginesize',
       'curbweight', 'curbweight', 'curbweight', 'dohcv', 'rwd', 'curbweight',
       'ohcf', 'carwidth', 'gas', 'turbo', 'rotor', 'carlength', 'horsepower',
       'Fueleconomy', 'fwd', 'curbweight', 'ohcv', 'sedan', 'curbweight'],
      dtype='object')
```

Out[525…

	curbweight	horsepower	Fueleconomy	carwidth	hatchback	sedan	wagon	dohcv	twelve	high
122	0.272692	0.083333	0.530864	0.291667	0	1	0	0	0	
125	0.500388	0.395833	0.213992	0.666667	1	0	0	0	0	
166	0.314973	0.266667	0.344307	0.308333	1	0	0	0	0	
1	0.411171	0.262500	0.244170	0.316667	0	0	0	0	0	
199	0.647401	0.475000	0.122085	0.575000	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	

	curbweight	horsepower	Fueleconomy	carwidth	hatchback	sedan	wagon	dohcv	twelve	high
87	0.354926	0.283333	0.311385	0.425000	0	1	0	0	0	
103	0.609775	0.433333	0.189300	0.516667	0	1	0	0	0	
67	0.786268	0.312500	0.234568	0.833333	0	1	0	0	0	
24	0.185803	0.083333	0.530864	0.291667	1	0	0	0	0	
8	0.619860	0.383333	0.097394	0.925000	0	1	0	0	0	

143 rows × 10 columns

building model using stats model for the detailed statistics

In [526…
```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor    #https://
```

In [527…
```
def build_model(X,y):
    X=sm.add_constant(X)
    print(X)
    lm=sm.OLS(y,X)
    a=lm.fit()
    print(a.params)   # constant is bias term and remaining are weights for each featur
    print(a.summary())
    return X
X_train_new=build_model(X_train_rfe,y_train)
print(X_train_new)
```

```
     const  curbweight  horsepower  Fueleconomy  carwidth  hatchback  sedan  \
122    1.0    0.272692    0.083333     0.530864  0.291667          0      1
125    1.0    0.500388    0.395833     0.213992  0.666667          1      0
166    1.0    0.314973    0.266667     0.344307  0.308333          1      0
1      1.0    0.411171    0.262500     0.244170  0.316667          0      0
199    1.0    0.647401    0.475000     0.122085  0.575000          0      0
..     ...         ...         ...          ...       ...        ...    ...
87     1.0    0.354926    0.283333     0.311385  0.425000          0      1
103    1.0    0.609775    0.433333     0.189300  0.516667          0      1
67     1.0    0.786268    0.312500     0.234568  0.833333          0      1
24     1.0    0.185803    0.083333     0.530864  0.291667          1      0
8      1.0    0.619860    0.383333     0.097394  0.925000          0      1

     wagon  dohcv  twelve  highend
122      0      0       0        0
125      0      0       0        1
166      0      0       0        0
1        0      0       0        0
199      1      0       0        0
..     ...    ...     ...      ...
87       0      0       0        0
103      0      0       0        0
67       0      0       0        1
24       0      0       0        0
8        0      0       0        0

[143 rows x 11 columns]
```

```
const         -0.094678
curbweight     0.265719
horsepower     0.449902
Fueleconomy    0.093304
carwidth       0.260878
hatchback     -0.092919
sedan         -0.070393
wagon         -0.099719
dohcv         -0.267605
twelve        -0.119244
highend        0.258633
dtype: float64
```

OLS Regression Results

Dep. Variable:	price	R-squared:	0.929
Model:	OLS	Adj. R-squared:	0.923
Method:	Least Squares	F-statistic:	172.1
Date:	Sun, 02 May 2021	Prob (F-statistic):	1.29e-70
Time:	00:44:56	Log-Likelihood:	205.85
No. Observations:	143	AIC:	-389.7
Df Residuals:	132	BIC:	-357.1
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0947	0.042	-2.243	0.027	-0.178	-0.011
curbweight	0.2657	0.069	3.870	0.000	0.130	0.402
horsepower	0.4499	0.074	6.099	0.000	0.304	0.596
Fueleconomy	0.0933	0.052	1.792	0.075	-0.010	0.196
carwidth	0.2609	0.062	4.216	0.000	0.138	0.383
hatchback	-0.0929	0.025	-3.707	0.000	-0.143	-0.043
sedan	-0.0704	0.025	-2.833	0.005	-0.120	-0.021
wagon	-0.0997	0.028	-3.565	0.001	-0.155	-0.044
dohcv	-0.2676	0.079	-3.391	0.001	-0.424	-0.112
twelve	-0.1192	0.067	-1.769	0.079	-0.253	0.014
highend	0.2586	0.020	12.929	0.000	0.219	0.298

Omnibus:	43.093	Durbin-Watson:	1.867
Prob(Omnibus):	0.000	Jarque-Bera (JB):	130.648
Skew:	1.128	Prob(JB):	4.27e-29
Kurtosis:	7.103	Cond. No.	32.0

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
     const  curbweight  horsepower  Fueleconomy  carwidth  hatchback  sedan  \
122    1.0    0.272692    0.083333     0.530864  0.291667          0      1
125    1.0    0.500388    0.395833     0.213992  0.666667          1      0
166    1.0    0.314973    0.266667     0.344307  0.308333          1      0
1      1.0    0.411171    0.262500     0.244170  0.316667          0      0
199    1.0    0.647401    0.475000     0.122085  0.575000          0      0
..     ...         ...         ...          ...       ...        ...    ...
87     1.0    0.354926    0.283333     0.311385  0.425000          0      1
103    1.0    0.609775    0.433333     0.189300  0.516667          0      1
67     1.0    0.786268    0.312500     0.234568  0.833333          0      1
24     1.0    0.185803    0.083333     0.530864  0.291667          1      0
8      1.0    0.619860    0.383333     0.097394  0.925000          0      1

     wagon  dohcv  twelve  highend
122      0      0       0        0
125      0      0       0        1
166      0      0       0        0
1        0      0       0        0
```

```
199      1      0       0        0
..     ...    ...     ...      ...
87       0      0       0        0
103      0      0       0        0
67       0      0       0        1
24       0      0       0        0
8        0      0       0        0

[143 rows x 11 columns]
```

Inference for above

- as p value > 0.05, then there is strong veidence for null hypothesis . so we will not consider those variable whose p >0.0.5
- twelve and fuel economy has p >0.05 , so drop those two independent variables

In [528…
```
X_train_new=X_train_new.drop(columns=["Fueleconomy","twelve"])
X_train_new
```

Out[528…

	const	curbweight	horsepower	carwidth	hatchback	sedan	wagon	dohcv	highend
122	1.0	0.272692	0.083333	0.291667	0	1	0	0	0
125	1.0	0.500388	0.395833	0.666667	1	0	0	0	1
166	1.0	0.314973	0.266667	0.308333	1	0	0	0	0
1	1.0	0.411171	0.262500	0.316667	0	0	0	0	0
199	1.0	0.647401	0.475000	0.575000	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...
87	1.0	0.354926	0.283333	0.425000	0	1	0	0	0
103	1.0	0.609775	0.433333	0.516667	0	1	0	0	0
67	1.0	0.786268	0.312500	0.833333	0	1	0	0	1
24	1.0	0.185803	0.083333	0.291667	1	0	0	0	0
8	1.0	0.619860	0.383333	0.925000	0	1	0	0	0

143 rows × 9 columns

In [529…
```
X_train_new=build_model(X_train_new,y_train)
```

```
     const  curbweight  horsepower  carwidth  hatchback  sedan  wagon  dohcv  \
122    1.0    0.272692    0.083333  0.291667          0      1      0      0
125    1.0    0.500388    0.395833  0.666667          1      0      0      0
166    1.0    0.314973    0.266667  0.308333          1      0      0      0
1      1.0    0.411171    0.262500  0.316667          0      0      0      0
199    1.0    0.647401    0.475000  0.575000          0      0      1      0
..     ...         ...         ...       ...        ...    ...    ...    ...
87     1.0    0.354926    0.283333  0.425000          0      1      0      0
103    1.0    0.609775    0.433333  0.516667          0      1      0      0
67     1.0    0.786268    0.312500  0.833333          0      1      0      0
24     1.0    0.185803    0.083333  0.291667          1      0      0      0
8      1.0    0.619860    0.383333  0.925000          0      1      0      0

     highend
```

```
122        0
125        1
166        0
1          0
199        0
..       ...
87         0
103        0
67         1
24         0
8          0

[143 rows x 9 columns]
const        -0.030528
curbweight    0.259330
horsepower    0.346912
carwidth      0.248768
hatchback    -0.092197
sedan        -0.071056
wagon        -0.104711
dohcv        -0.196832
highend       0.261046
dtype: float64
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.926
Model:                            OLS   Adj. R-squared:                  0.922
Method:                 Least Squares   F-statistic:                     209.5
Date:                Sun, 02 May 2021   Prob (F-statistic):           7.85e-72
Time:                        00:44:56   Log-Likelihood:                 203.07
No. Observations:                 143   AIC:                            -388.1
Df Residuals:                     134   BIC:                            -361.5
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0305      0.026     -1.165      0.246      -0.082       0.021
curbweight     0.2593      0.068      3.796      0.000       0.124       0.394
horsepower     0.3469      0.058      5.964      0.000       0.232       0.462
carwidth       0.2488      0.062      3.995      0.000       0.126       0.372
hatchback     -0.0922      0.025     -3.650      0.000      -0.142      -0.042
sedan         -0.0711      0.025     -2.850      0.005      -0.120      -0.022
wagon         -0.1047      0.028     -3.721      0.000      -0.160      -0.049
dohcv         -0.1968      0.073     -2.689      0.008      -0.342      -0.052
highend        0.2610      0.020     13.083      0.000       0.222       0.301
==============================================================================
Omnibus:                       48.637   Durbin-Watson:                   1.909
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              161.444
Skew:                           1.250   Prob(JB):                     8.77e-36
Kurtosis:                       7.566   Cond. No.                         27.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In [530…

```
# VIF detects multicollinerairty in regression analysis
# here multicollinearoty refers to correlation b/w independent variables in data  which
def Detect_Multicollinearity(X):
    vif=pd.DataFrame()
    vif['features']=X.columns
    print(X.columns,vif['features'])
    vif['VIF']=[variance_inflation_factor(X.values,i) for i in range(0,X.shape[1])]
```

```
    vif['VIF']=round(vif['VIF'],3)# upto 3 decimal points
    vif=vif.sort_values(['VIF'],ascending=[False])
    return vif
Detect_Multicollinearity(X_train_new)
```

```
Index(['const', 'curbweight', 'horsepower', 'carwidth', 'hatchback', 'sedan',
       'wagon', 'dohcv', 'highend'],
      dtype='object') 0         const
1     curbweight
2     horsepower
3       carwidth
4      hatchback
5          sedan
6          wagon
7          dohcv
8        highend
Name: features, dtype: object
```

Out[530…

	features	VIF
0	const	26.897
1	curbweight	8.104
5	sedan	6.073
4	hatchback	5.630
3	carwidth	5.136
2	horsepower	3.605
6	wagon	3.575
8	highend	1.634
7	dohcv	1.458

In [531…

```
Detect_Multicollinearity(X_train_new)
X_train_new=X_train_new.drop(columns=['curbweight'])  # dropping curb weight because of
X_train_new
```

```
Index(['const', 'curbweight', 'horsepower', 'carwidth', 'hatchback', 'sedan',
       'wagon', 'dohcv', 'highend'],
      dtype='object') 0         const
1     curbweight
2     horsepower
3       carwidth
4      hatchback
5          sedan
6          wagon
7          dohcv
8        highend
Name: features, dtype: object
```

Out[531…

	const	horsepower	carwidth	hatchback	sedan	wagon	dohcv	highend
122	1.0	0.083333	0.291667	0	1	0	0	0
125	1.0	0.395833	0.666667	1	0	0	0	1
166	1.0	0.266667	0.308333	1	0	0	0	0
1	1.0	0.262500	0.316667	0	0	0	0	0
199	1.0	0.475000	0.575000	0	0	1	0	0

	const	horsepower	carwidth	hatchback	sedan	wagon	dohcv	highend
...	...	...	...	...	...	...	...	...
87	1.0	0.283333	0.425000	0	1	0	0	0
103	1.0	0.433333	0.516667	0	1	0	0	0
67	1.0	0.312500	0.833333	0	1	0	0	1
24	1.0	0.083333	0.291667	1	0	0	0	0
8	1.0	0.383333	0.925000	0	1	0	0	0

143 rows × 8 columns

In [532…
```
X_train_new=build_model(X_train_new,y_train)
```

```
     const  horsepower  carwidth  hatchback  sedan  wagon  dohcv  highend
122    1.0    0.083333  0.291667          0      1      0      0        0
125    1.0    0.395833  0.666667          1      0      0      0        1
166    1.0    0.266667  0.308333          1      0      0      0        0
1      1.0    0.262500  0.316667          0      0      0      0        0
199    1.0    0.475000  0.575000          0      0      1      0        0
..     ...         ...       ...        ...    ...    ...    ...      ...
87     1.0    0.283333  0.425000          0      1      0      0        0
103    1.0    0.433333  0.516667          0      1      0      0        0
67     1.0    0.312500  0.833333          0      1      0      0        1
24     1.0    0.083333  0.291667          1      0      0      0        0
8      1.0    0.383333  0.925000          0      1      0      0        0

[143 rows x 8 columns]
const        -0.031901
horsepower    0.468956
carwidth      0.426864
hatchback    -0.104434
sedan        -0.075606
wagon        -0.086462
dohcv        -0.310561
highend       0.277158
dtype: float64
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.918
Model:                            OLS   Adj. R-squared:                  0.914
Method:                 Least Squares   F-statistic:                     215.9
Date:                Sun, 02 May 2021   Prob (F-statistic):           4.70e-70
Time:                        00:44:57   Log-Likelihood:                 195.77
No. Observations:                 143   AIC:                            -375.5
Df Residuals:                     135   BIC:                            -351.8
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0319      0.027     -1.161      0.248      -0.086       0.022
horsepower     0.4690      0.051      9.228      0.000       0.368       0.569
carwidth       0.4269      0.043      9.944      0.000       0.342       0.512
hatchback     -0.1044      0.026     -3.976      0.000      -0.156      -0.052
sedan         -0.0756      0.026     -2.896      0.004      -0.127      -0.024
wagon         -0.0865      0.029     -2.974      0.003      -0.144      -0.029
dohcv         -0.3106      0.070     -4.435      0.000      -0.449      -0.172
highend        0.2772      0.020     13.559      0.000       0.237       0.318
==============================================================================
```

```
Omnibus:                       43.937   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              127.746
Skew:                           1.171   Prob(JB):                     1.82e-28
Kurtosis:                       6.995   Cond. No.                         18.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In [533…
```
Detect_Multicollinearity(X_train_new)
```

```
Index(['const', 'horsepower', 'carwidth', 'hatchback', 'sedan', 'wagon',
       'dohcv', 'highend'],
      dtype='object') 0         const
1    horsepower
2      carwidth
3     hatchback
4         sedan
5         wagon
6         dohcv
7       highend
Name: features, dtype: object
```

Out[533…

	features	VIF
0	const	26.891
4	sedan	6.059
3	hatchback	5.539
5	wagon	3.471
1	horsepower	2.504
2	carwidth	2.220
7	highend	1.560
6	dohcv	1.214

In [534…
```
# Drop sedan becuase of high vif value
X_train_new=X_train_new.drop('sedan', axis=1)
X_train_new
```

Out[534…

	const	horsepower	carwidth	hatchback	wagon	dohcv	highend
122	1.0	0.083333	0.291667	0	0	0	0
125	1.0	0.395833	0.666667	1	0	0	1
166	1.0	0.266667	0.308333	1	0	0	0
1	1.0	0.262500	0.316667	0	0	0	0
199	1.0	0.475000	0.575000	0	1	0	0
…	…	…	…	…	…	…	…
87	1.0	0.283333	0.425000	0	0	0	0
103	1.0	0.433333	0.516667	0	0	0	0
67	1.0	0.312500	0.833333	0	0	0	1

	const	horsepower	carwidth	hatchback	wagon	dohcv	highend
24	1.0	0.083333	0.291667	1	0	0	0
8	1.0	0.383333	0.925000	0	0	0	0

143 rows × 7 columns

In [535…
```
Detect_Multicollinearity(X_train_new)
```

```
Index(['const', 'horsepower', 'carwidth', 'hatchback', 'wagon', 'dohcv',
       'highend'],
      dtype='object') 0         const
1    horsepower
2      carwidth
3     hatchback
4         wagon
5         dohcv
6       highend
Name: features, dtype: object
```

Out[535…

	features	VIF
0	const	10.823
1	horsepower	2.392
2	carwidth	2.086
6	highend	1.554
3	hatchback	1.231
5	dohcv	1.211
4	wagon	1.110

In [536…
```
X_train_new=build_model(X_train_new,y_train)
```

```
     const  horsepower  carwidth  hatchback  wagon  dohcv  highend
122    1.0    0.083333  0.291667          0      0      0        0
125    1.0    0.395833  0.666667          1      0      0        1
166    1.0    0.266667  0.308333          1      0      0        0
1      1.0    0.262500  0.316667          0      0      0        0
199    1.0    0.475000  0.575000          0      1      0        0
..     ...         ...       ...        ...    ...    ...      ...
87     1.0    0.283333  0.425000          0      0      0        0
103    1.0    0.433333  0.516667          0      0      0        0
67     1.0    0.312500  0.833333          0      0      0        1
24     1.0    0.083333  0.291667          1      0      0        0
8      1.0    0.383333  0.925000          0      0      0        0

[143 rows x 7 columns]
const        -0.093391
horsepower    0.500066
carwidth      0.396303
hatchback    -0.037347
wagon        -0.017022
dohcv        -0.320316
highend       0.280767
dtype: float64
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                  price   R-squared:                       0.913
Model:                            OLS   Adj. R-squared:                  0.909
Method:                 Least Squares   F-statistic:                     237.6
Date:                Sun, 02 May 2021   Prob (F-statistic):           1.68e-69
Time:                        00:44:57   Log-Likelihood:                 191.46
No. Observations:                 143   AIC:                            -368.9
Df Residuals:                     136   BIC:                            -348.2
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const         -0.0934      0.018     -5.219      0.000      -0.129      -0.058
horsepower     0.5001      0.051      9.805      0.000       0.399       0.601
carwidth       0.3963      0.043      9.275      0.000       0.312       0.481
hatchback     -0.0373      0.013     -2.938      0.004      -0.062      -0.012
wagon         -0.0170      0.017     -1.008      0.315      -0.050       0.016
dohcv         -0.3203      0.072     -4.460      0.000      -0.462      -0.178
highend        0.2808      0.021     13.402      0.000       0.239       0.322
==============================================================================
Omnibus:                       34.143   Durbin-Watson:                   2.024
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               72.788
Skew:                           1.018   Prob(JB):                     1.56e-16
Kurtosis:                       5.841   Cond. No.                         16.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In [537…

```
#drop wagon because of high p value   >>> 0.05
X_train_new=X_train_new.drop(columns=['wagon'])
X_train_new
```

Out[537…

	const	horsepower	carwidth	hatchback	dohcv	highend
122	1.0	0.083333	0.291667	0	0	0
125	1.0	0.395833	0.666667	1	0	1
166	1.0	0.266667	0.308333	1	0	0
1	1.0	0.262500	0.316667	0	0	0
199	1.0	0.475000	0.575000	0	0	0
…	…	…	…	…	…	…
87	1.0	0.283333	0.425000	0	0	0
103	1.0	0.433333	0.516667	0	0	0
67	1.0	0.312500	0.833333	0	0	1
24	1.0	0.083333	0.291667	1	0	0
8	1.0	0.383333	0.925000	0	0	0

143 rows × 6 columns

In [538…

```
Detect_Multicollinearity(X_train_new)
```

```
Index(['const', 'horsepower', 'carwidth', 'hatchback', 'dohcv', 'highend'], dtype='objec
t') 0         const
```

```
1    horsepower
2      carwidth
3     hatchback
4          dohcv
5       highend
Name: features, dtype: object
```

Out[538…

	features	VIF
0	const	10.394
1	horsepower	2.390
2	carwidth	2.085
5	highend	1.532
4	dohcv	1.209
3	hatchback	1.127

Residual analysis of Model

In [539…

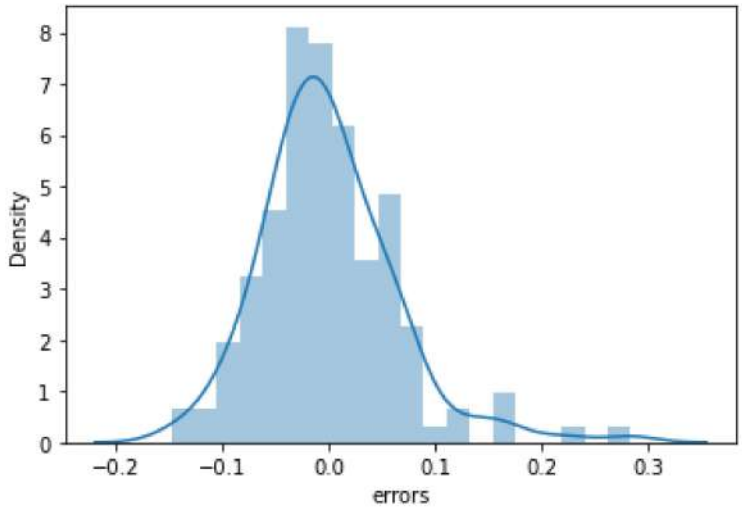
```
lm=sm.OLS(y_train,X_train_new).fit()
print(lm)
y_train_price=lm.predict(X_train_new)
print(y_train_price)
```

```
<statsmodels.regression.linear_model.RegressionResultsWrapper object at 0x00000167B8D162
20>
122    0.060063
125    0.614558
166    0.124929
1      0.159767
199    0.368396
         ...
87     0.213025
103    0.324453
67     0.672279
24     0.026428
8      0.460757
Length: 143, dtype: float64
```

In [540…

```
plt.figure()
#print(y_train-y_train_price)
sns.distplot((y_train-y_train_price),bins=20)  #20 equal intervals
plt.xlabel('errors')
plt.show()
```

```
C:\Users\prasa\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [541…
```
# errors are normally distributed and the assumption of linear modelling seems to be fu
```

Prediction and Evaluation

In [542…
```
print(df_test)
```

```
     price  wheelbase  curbweight  enginesize  boreratio  horsepower  \
160   7738       95.7        2094          98       3.19          70
186   8495       97.3        2275         109       3.19          85
59    8845       98.8        2385         122       3.39          84
165   9298       94.5        2265          98       3.24         112
140   7603       93.3        2240         108       3.62          73
..     ...        ...         ...         ...        ...         ...
28    8921      103.3        2535         122       3.34          88
29   12964       95.9        2811         156       3.60         145
182   7775       97.3        2261          97       3.01          52
40   10295       96.5        2372         110       3.15          86
128  37028       89.5        2800         194       3.74         207

     Fueleconomy  carlength  carwidth  gas  ...  ohcv  rotor  five  four  six  \
160        42.05      166.3      64.4    1  ...     0      0     0     1    0
186        30.15      171.7      65.5    1  ...     0      0     0     1    0
59         28.70      177.8      66.5    1  ...     0      0     0     1    0
165        27.35      168.7      64.0    1  ...     0      0     0     1    0
140        28.25      157.3      63.8    1  ...     0      0     0     1    0
..           ...        ...       ...  ...  ...   ...    ...   ...   ...  ...
28         26.70      174.6      64.6    1  ...     0      0     0     1    0
29         21.25      173.2      66.3    1  ...     0      0     0     1    0
182        41.05      171.7      65.5    0  ...     0      0     0     1    0
40         29.70      175.4      62.5    1  ...     0      0     0     1    0
128        20.60      168.9      65.0    1  ...     0      0     0     0    1

     three  twelve  two  medium  highend
160      0       0    0       0        0
186      0       0    0       1        0
59       0       0    0       1        0
165      0       0    0       0        0
140      0       0    0       0        0
..     ...     ...  ...     ...      ...
28       0       0    0       0        0
29       0       0    0       0        0
182      0       0    0       1        0
40       0       0    0       0        0
128      0       0    0       0        1
```

```
[62 rows x 31 columns]
```

In [543…
```
X_train_new
```

Out[543…

	const	horsepower	carwidth	hatchback	dohcv	highend
122	1.0	0.083333	0.291667	0	0	0
125	1.0	0.395833	0.666667	1	0	1
166	1.0	0.266667	0.308333	1	0	0
1	1.0	0.262500	0.316667	0	0	0
199	1.0	0.475000	0.575000	0	0	0
…	…	…	…	…	…	…
87	1.0	0.283333	0.425000	0	0	0
103	1.0	0.433333	0.516667	0	0	0
67	1.0	0.312500	0.833333	0	0	1
24	1.0	0.083333	0.291667	1	0	0
8	1.0	0.383333	0.925000	0	0	0

143 rows × 6 columns

In [544…
```
#X_train_new=X_train_new.drop(columns=['const'])
#X_train_new

#df_test=scaler.fit_transform(df_test)
#df_test
```

In [545…
```
#X_test=df_test[X_train_new(columns=['horsepower','carwidth','hatchback','dohcv','highe
#print(X_test)
#X_test_new=scaler.fit_transform(X_test)#(columns=['horsepower','carwidth','hatchback',
#X_test_new

num_vars=['horsepower','carwidth','hatchback','dohcv','highend','price']
df_test[num_vars]=scaler.fit_transform(df_test[num_vars])
df_test
```

```
<ipython-input-545-cd2a31e1d6bd>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df_test[num_vars]=scaler.fit_transform(df_test[num_vars])
C:\Users\prasa\anaconda3\lib\site-packages\pandas\core\indexing.py:1736: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value[:, i].tolist())
```

Out[545…

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	Fueleconomy	carlength	car

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	Fueleconomy	carlength	car
160	0.058474	95.7	2094	98	3.19	0.116129	42.05	166.3	0.2
186	0.077398	97.3	2275	109	3.19	0.212903	30.15	171.7	0.3
59	0.086148	98.8	2385	122	3.39	0.206452	28.70	177.8	0.4
165	0.097473	94.5	2265	98	3.24	0.387097	27.35	168.7	0.1
140	0.055099	93.3	2240	108	3.62	0.135484	28.25	157.3	0.1
...	...	...	...	...	...	...	...	...	
28	0.088048	103.3	2535	122	3.34	0.232258	26.70	174.6	0.2
29	0.189120	95.9	2811	156	3.60	0.600000	21.25	173.2	0.4
182	0.059399	97.3	2261	97	3.01	0.000000	41.05	171.7	0.3
40	0.122397	96.5	2372	110	3.15	0.219355	29.70	175.4	0.0
128	0.790705	89.5	2800	194	3.74	1.000000	20.60	168.9	0.2

62 rows × 31 columns

In [546…
```
y_test=df_test.pop('price')
print(y_test)
```

```
160    0.058474
186    0.077398
59     0.086148
165    0.097473
140    0.055099
         ...
28     0.088048
29     0.189120
182    0.059399
40     0.122397
128    0.790705
Name: price, Length: 62, dtype: float64
```

In [547…
```
#X_test_new=scaler.fit_transform(X_test)
#print(X_test_new)

X_test=df_test
print(X_test)
```

```
     wheelbase  curbweight  enginesize  boreratio  horsepower  Fueleconomy  \
160       95.7        2094          98       3.19    0.116129        42.05
186       97.3        2275         109       3.19    0.212903        30.15
59        98.8        2385         122       3.39    0.206452        28.70
165       94.5        2265          98       3.24    0.387097        27.35
140       93.3        2240         108       3.62    0.135484        28.25
..         ...         ...         ...        ...         ...          ...
28       103.3        2535         122       3.34    0.232258        26.70
29        95.9        2811         156       3.60    0.600000        21.25
182       97.3        2261          97       3.01    0.000000        41.05
40        96.5        2372         110       3.15    0.219355        29.70
128       89.5        2800         194       3.74    1.000000        20.60

     carlength  carwidth  gas  turbo  ...  ohcv  rotor  five  four  six  \
160      166.3  0.200000    1      0  ...     0      0     0     1    0
```

```
186      171.7  0.315789    1      0  ...     0      0      0     1    0
59       177.8  0.421053    1      0  ...     0      0      0     1    0
165      168.7  0.157895    1      0  ...     0      0      0     1    0
140      157.3  0.136842    1      0  ...     0      0      0     1    0
..         ...       ...  ...    ...  ...   ...    ...    ...   ...  ...
28       174.6  0.221053    1      0  ...     0      0      0     1    0
29       173.2  0.400000    1      1  ...     0      0      0     1    0
182      171.7  0.315789    0      0  ...     0      0      0     1    0
40       175.4  0.000000    1      0  ...     0      0      0     1    0
128      168.9  0.263158    1      0  ...     0      0      0     0    1

     three  twelve  two  medium  highend
160      0       0    0       0      0.0
186      0       0    0       1      0.0
59       0       0    0       1      0.0
165      0       0    0       0      0.0
140      0       0    0       0      0.0
..     ...     ...  ...     ...      ...
28       0       0    0       0      0.0
29       0       0    0       0      0.0
182      0       0    0       1      0.0
40       0       0    0       0      0.0
128      0       0    0       0      1.0

[62 rows x 30 columns]
```

In [548…

```
X_train_new=X_train_new.drop("const",axis=1)    #  (columns=['const'])
X_train_new
```

Out[548…

	horsepower	carwidth	hatchback	dohcv	highend
122	0.083333	0.291667	0	0	0
125	0.395833	0.666667	1	0	1
166	0.266667	0.308333	1	0	0
1	0.262500	0.316667	0	0	0
199	0.475000	0.575000	0	0	0
…	…	…	…	…	…
87	0.283333	0.425000	0	0	0
103	0.433333	0.516667	0	0	0
67	0.312500	0.833333	0	0	1
24	0.083333	0.291667	1	0	0
8	0.383333	0.925000	0	0	0

143 rows × 5 columns

In [549…

```
X_test_new=X_test[X_train_new.columns]
X_test_new
```

Out[549…

	horsepower	carwidth	hatchback	dohcv	highend
160	0.116129	0.200000	0.0	0.0	0.0
186	0.212903	0.315789	0.0	0.0	0.0

	horsepower	carwidth	hatchback	dohcv	highend
59	0.206452	0.421053	1.0	0.0	0.0
165	0.387097	0.157895	0.0	0.0	0.0
140	0.135484	0.136842	1.0	0.0	0.0
...	...	...	...	...	...
28	0.232258	0.221053	0.0	0.0	0.0
29	0.600000	0.400000	1.0	0.0	0.0
182	0.000000	0.315789	0.0	0.0	0.0
40	0.219355	0.000000	0.0	0.0	0.0
128	1.000000	0.263158	0.0	0.0	1.0

62 rows × 5 columns

In [550…
```
X_test_new=sm.add_constant(X_test_new)
X_test_new
```

Out[550…

	const	horsepower	carwidth	hatchback	dohcv	highend
160	1.0	0.116129	0.200000	0.0	0.0	0.0
186	1.0	0.212903	0.315789	0.0	0.0	0.0
59	1.0	0.206452	0.421053	1.0	0.0	0.0
165	1.0	0.387097	0.157895	0.0	0.0	0.0
140	1.0	0.135484	0.136842	1.0	0.0	0.0
...	...	...	...	...	...	...
28	1.0	0.232258	0.221053	0.0	0.0	0.0
29	1.0	0.600000	0.400000	1.0	0.0	0.0
182	1.0	0.000000	0.315789	0.0	0.0	0.0
40	1.0	0.219355	0.000000	0.0	0.0	0.0
128	1.0	1.000000	0.263158	0.0	0.0	1.0

62 rows × 6 columns

In [551…
```
y_pred=lm.predict(X_test_new)
print(y_pred)
```

```
160    0.040278
186    0.134555
59     0.139286
165    0.159488
140   -0.008612
        ...
28     0.106819
29     0.328271
182    0.027817
```

```
40     0.012992
128    0.791626
Length: 62, dtype: float64
```

In [552…
```
print(y_pred,y_test)
```

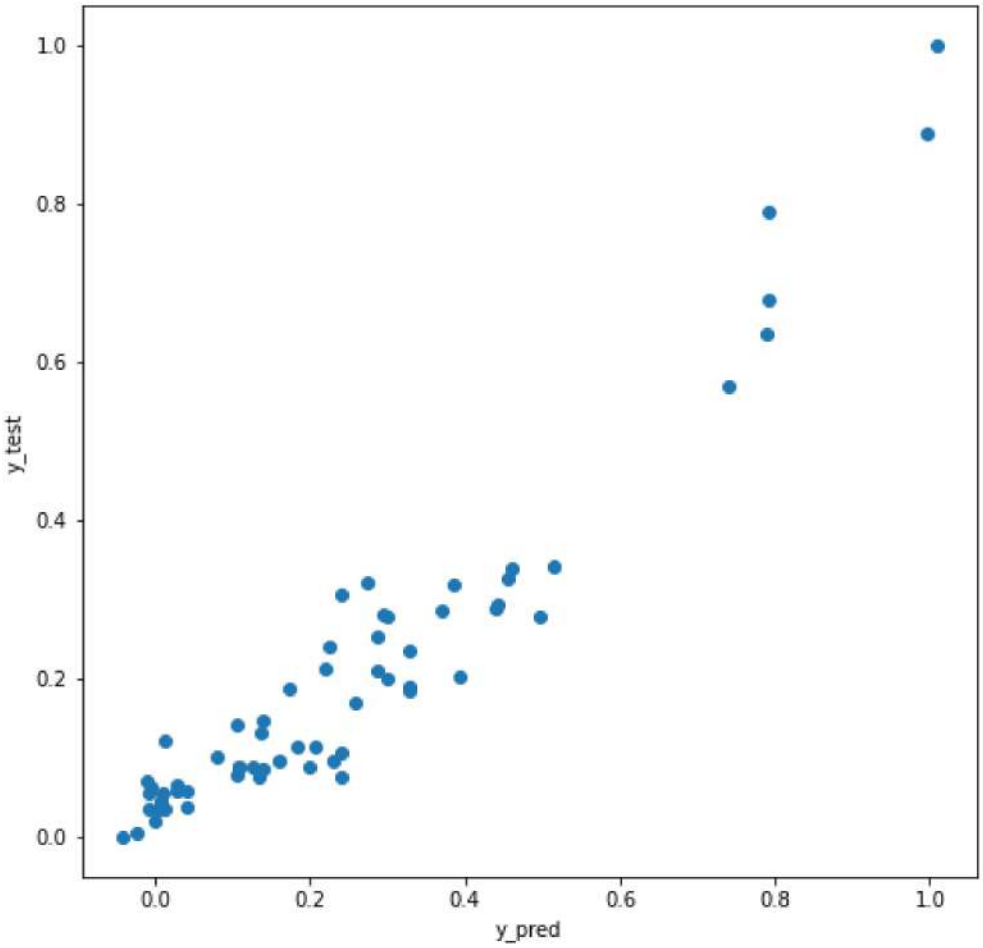
```
160    0.040278
186    0.134555
59     0.139286
165    0.159488
140   -0.008612
         ...
28     0.106819
29     0.328271
182    0.027817
40     0.012992
128    0.791626
Length: 62, dtype: float64 160    0.058474
186    0.077398
59     0.086148
165    0.097473
140    0.055099
         ...
28     0.088048
29     0.189120
182    0.059399
40     0.122397
128    0.790705
Name: price, Length: 62, dtype: float64
```

In [553…
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)    # r2_score(y_pred,y_test)  will give 0.86 which is false  be
#the syntax of r2 score function contains first argument as true and second argument as
```

Out[553…
```
0.8146387226353737
```

In [554…
```
plt.figure(figsize=(8,8))
plt.xlabel("y_pred")
plt.ylabel("y_test")
plt.scatter(y_pred,y_test)
#plt.plot(y_pred,y_test,)#color='red')
```

Out[554…
```
<matplotlib.collections.PathCollection at 0x167aef59580>
```



In [555…
```
print(lm.summary())
```

OLS Regression Results

Dep. Variable:	price	R-squared:	0.912
Model:	OLS	Adj. R-squared:	0.909
Method:	Least Squares	F-statistic:	284.8
Date:	Sun, 02 May 2021	Prob (F-statistic):	1.57e-70
Time:	00:44:57	Log-Likelihood:	190.93
No. Observations:	143	AIC:	-369.9
Df Residuals:	137	BIC:	-352.1
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0970	0.018	-5.530	0.000	-0.132	-0.062
horsepower	0.5013	0.051	9.832	0.000	0.401	0.602
carwidth	0.3952	0.043	9.252	0.000	0.311	0.480
hatchback	-0.0336	0.012	-2.764	0.006	-0.058	-0.010
dohcv	-0.3231	0.072	-4.502	0.000	-0.465	-0.181
highend	0.2833	0.021	13.615	0.000	0.242	0.324

Omnibus:	36.097	Durbin-Watson:	2.028
Prob(Omnibus):	0.000	Jarque-Bera (JB):	78.717
Skew:	1.067	Prob(JB):	8.07e-18
Kurtosis:	5.943	Cond. No.	16.3

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.

In [556…

```
# all p values are less than 0.05
```