In [ ]:

```
#### Problem Statement:
A Chinese automobile company Geely Auto aspires to enter the US market by setting up their manufact
uring unit there and
producing cars locally to give competition to their US and European counterparts.

They have contracted an automobile consulting company to understand the factors on which the
pricing of cars depends.
Specifically, they want to understand the factors affecting the pricing of cars in the American
market, since those may be very different from the Chinese market. The company wants to know:

Which variables are significant in predicting the price of a car
How well those variables describe the price of a car   #########

###########  Business Goal
We are required to model the price of cars with the available independent variables.
It will be used by the management to understand how exactly the prices vary with the independent va
riables.
They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain
price levels.
Further, the model will be a good way for management to understand the pricing dynamics of a new ma
rket. #########

###### table of content
Step 1: Reading and Understanding the Data
Step 2: Cleaning the Data
Missing Value check
Data type check
Duplicate check
Step 3: Data Visualization
Boxplot
Pairplot
Step 4: Data Preparation
Dummy Variable
Step 5: Splitting the Data into Training and Testing Sets
Rescaling
Step 6: Building a Linear Model
RFE
VIF
Step 7: Residual Analysis of the train data
Step 8: Making Predictions Using the Final Model
Step 9: Model Evaluation
RMSE Score  ###################################################
```

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as ticker   ####### for setting a tick for every integer multiple of a ba
se within the view interval.##
import matplotlib.ticker as plticker
import warnings
warnings.filterwarnings("ignore")
from datetime import datetime,timedelta
```

In [2]:

```
####### importing machine learning libraries #########

from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.base import TransformerMixin    ##### Mixin class for all transformers in scikit-
learn.#####\
from sklearn.preprocessing import MinMaxScaler
import statsmodels.api as sm
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import r2_score
```

In [3]:

```
car= pd.read_csv("carprice_data.csv")
```

In [4]:

```
car
```

Out[4]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesi
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	13
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	13
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	15
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	10
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	13
...	...	...	...	...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	14
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	14
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	17
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	14
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	14

205 rows × 26 columns

In [5]:

```
car.head()
```

Out[5]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136

5 rows × 26 columns

In [5]:

```
car.columns
```

Out[5]:

```
Index(['car_ID', 'symboling', 'CarName', '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')
```

In [6]:

```
car.shape
```

Out[6]:

```
(205, 26)
```

In [7]:

```
car.info()
```

```
<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
```

In [8]:

```
car.describe()
```

Out[8]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	com
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	

In [9]:

```
car.isnull().sum() #### checking null value #####
```

Out[9]:

Out[9]:

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

In [10]:

```
car.isna().mean() ###### checking missing value########
```

Out[10]:

```
car_ID              0.0
symboling           0.0
CarName             0.0
fueltype            0.0
aspiration          0.0
doornumber          0.0
carbody             0.0
drivewheel          0.0
enginelocation      0.0
wheelbase           0.0
carlength           0.0
carwidth            0.0
carheight           0.0
curbweight          0.0
enginetype          0.0
cylindernumber      0.0
enginesize          0.0
fuelsystem          0.0
boreratio           0.0
stroke              0.0
compressionratio    0.0
horsepower          0.0
peakrpm             0.0
citympg             0.0
highwaympg          0.0
price               0.0
dtype: float64
```

In [11]:

```
###### dropping the ID column #######
car = car.drop('car_ID',axis=1)
```

In [12]:

```
# Extracting Car Company from the CarName as per direction in Problem

car['CarName'] = car['CarName'].str.split(' ',expand=True)
```

In [13]:

```
##### unique car company

car['CarName'].unique()
```

Out[13]:

```
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 [ ]:

```
#######A typo is then any bug which is the direct result of mistyping part of the code
######Typo Error in Car Company name

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

In [14]:

```
#### renaming the typo error in car company names ######

car['CarName']=car['CarName'].replace({'maxda':'mazda','nissan':'Nissan','porcshce':'porsche','toyc
uta':'toyota','vokswagen':'volkswagen','vw':'volkswagen'})
```

In [15]:

```
# changing the datatype of symboling

car['symboling'] = car['symboling'].astype(str)
```

In [16]:

```
car['symboling'].head(10)
```

Out[16]:

```
0    3
1    3
2    1
3    2
4    2
5    2
6    1
7    1
8    1
9    0
Name: symboling, dtype: object
```

In [17]:

```
car['symboling'].tail(20)
```

Out[17]:

```
185    2
186    2
187    2
188    2
189    3
190    3
191    0
192    0
```

```
193     0
194    -2
195    -1
196    -2
197    -1
198    -2
199    -1
200    -1
201    -1
202    -1
203    -1
204    -1
Name: symboling, dtype: object
```

In [18]:

```
######### checking duplicates ###########
car.loc[car.duplicated()]
```

Out[18]:

symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...	enginesize	f

0 rows × 25 columns

In [19]:

```
######### segregation of categorical and numerical variable #########

cat_col=car.select_dtypes(include=['object']).columns
num_col=car.select_dtypes(exclude=['object']).columns
df_cat=car[cat_col]
df_num=car[num_col]
```

In [20]:

```
df_cat.head(10)
```

Out[20]:

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype	cylindernumber	fuelsyst
0	3	alfa-romero	gas	std	two	convertible	rwd	front	dohc	four	r
1	3	alfa-romero	gas	std	two	convertible	rwd	front	dohc	four	r
2	1	alfa-romero	gas	std	two	hatchback	rwd	front	ohcv	six	r
3	2	audi	gas	std	four	sedan	fwd	front	ohc	four	r
4	2	audi	gas	std	four	sedan	4wd	front	ohc	five	r
5	2	audi	gas	std	two	sedan	fwd	front	ohc	five	r
6	1	audi	gas	std	four	sedan	fwd	front	ohc	five	r
7	1	audi	gas	std	four	wagon	fwd	front	ohc	five	r
8	1	audi	gas	turbo	four	sedan	fwd	front	ohc	five	r
9	0	audi	gas	turbo	two	hatchback	4wd	front	ohc	five	r

In [21]:

```
df_num.head(10)
```

Out[21]:

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	cityn
0	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	111	5000	

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citym
1	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	9.0	111	5000	
2	94.5	171.2	65.5	52.4	2823	152	2.68	3.47	9.0	154	5000	
3	99.8	176.6	66.2	54.3	2337	109	3.19	3.40	10.0	102	5500	
4	99.4	176.6	66.4	54.3	2824	136	3.19	3.40	8.0	115	5500	
5	99.8	177.3	66.3	53.1	2507	136	3.19	3.40	8.5	110	5500	
6	105.8	192.7	71.4	55.7	2844	136	3.19	3.40	8.5	110	5500	
7	105.8	192.7	71.4	55.7	2954	136	3.19	3.40	8.5	110	5500	
8	105.8	192.7	71.4	55.9	3086	131	3.13	3.40	8.3	140	5500	
9	99.5	178.2	67.9	52.0	3053	131	3.13	3.40	7.0	160	5500	

In [22]:

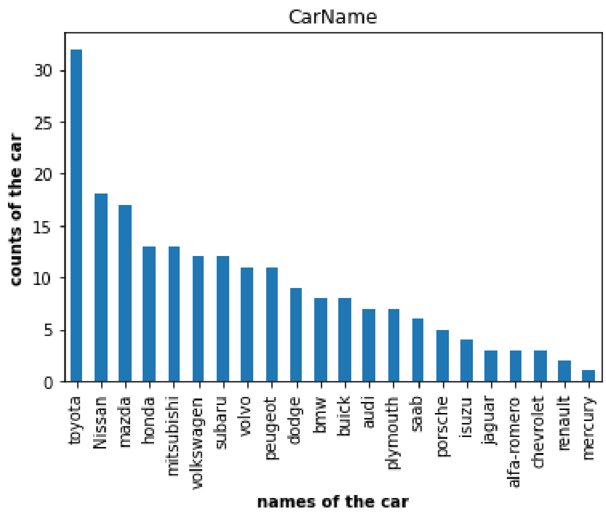
```
###########visualization ##############

car['CarName'].value_counts()
```

Out[22]:

```
toyota         32
Nissan         18
mazda          17
honda          13
mitsubishi     13
volkswagen     12
subaru         12
volvo          11
peugeot        11
dodge           9
bmw             8
buick           8
audi            7
plymouth        7
saab            6
porsche         5
isuzu           4
jaguar          3
alfa-romero     3
chevrolet       3
renault         2
mercury         1
Name: CarName, dtype: int64
```

In [23]:

```
ax=car['CarName'].value_counts().plot(kind='bar')
ax.title.set_text('CarName')
plt.xlabel('names of the car',fontweight='bold')
plt.ylabel('counts of the car',fontweight='bold')
plt.show()
```

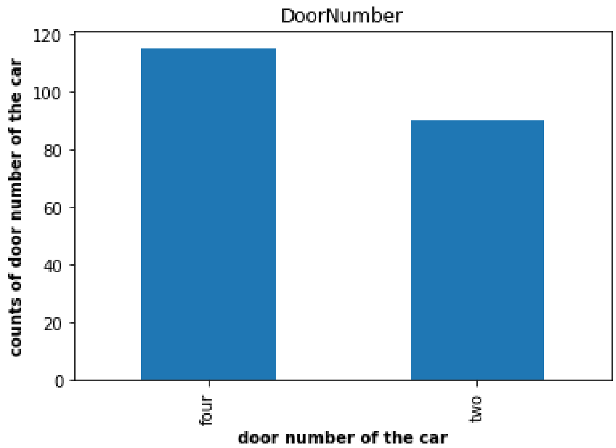


In [ ]:

```
#######Insights:
#######Toyota seems to be the most favoured cars.
#######Mercury seems to be the least favoured cars.
```

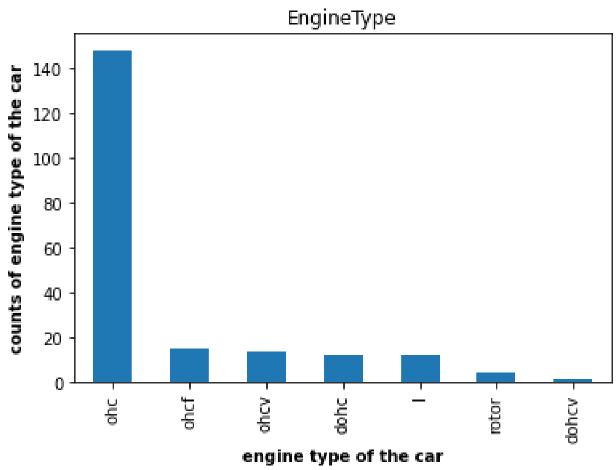
In [24]:

```
ax=car['doornumber'].value_counts().plot(kind='bar')
ax.title.set_text('DoorNumber')
plt.xlabel('door number of the car',fontweight='bold')
plt.ylabel('counts of door number of the car',fontweight='bold')
plt.show()
```



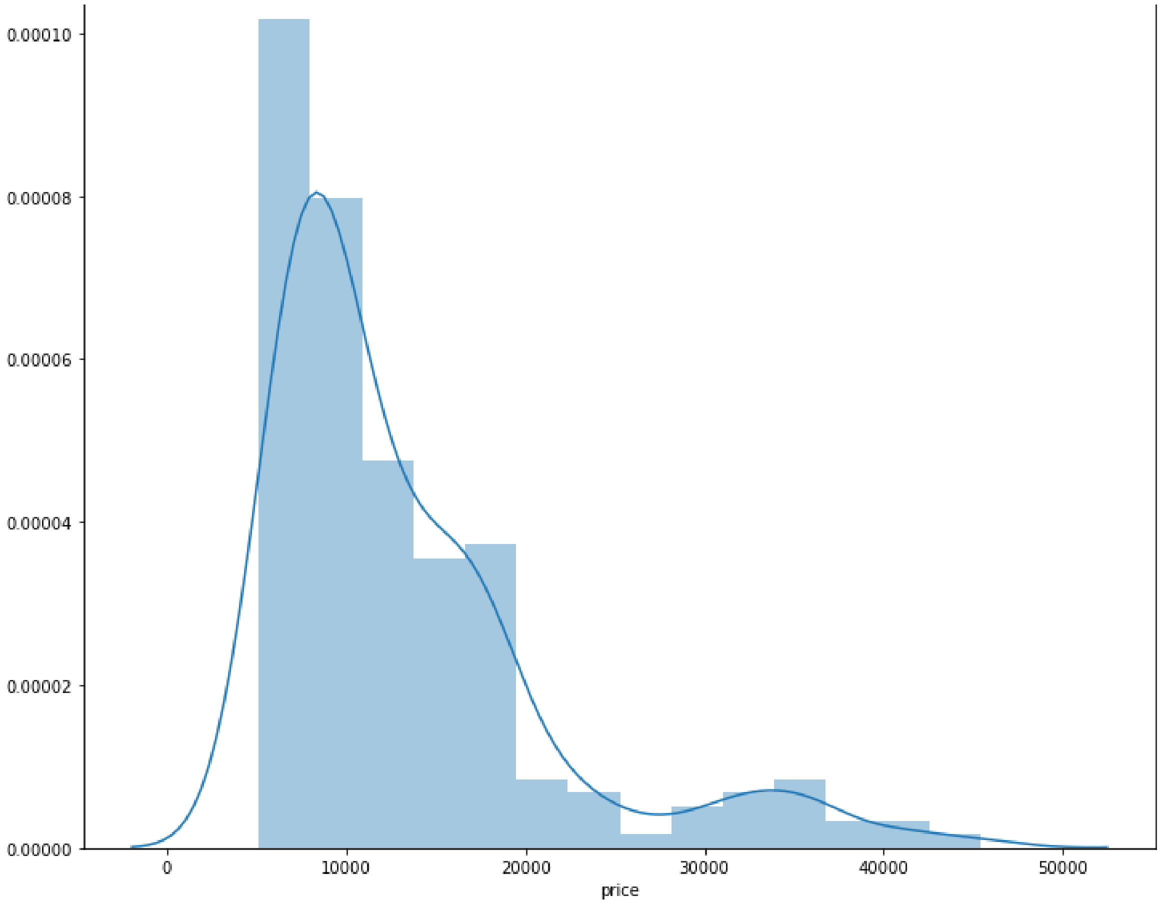
In [25]:

```
ax=car['enginetype'].value_counts().plot(kind='bar')
ax.title.set_text('EngineType')
plt.xlabel('engine type of the car',fontweight='bold')
plt.ylabel('counts of engine type of the car',fontweight='bold')
plt.show()
```



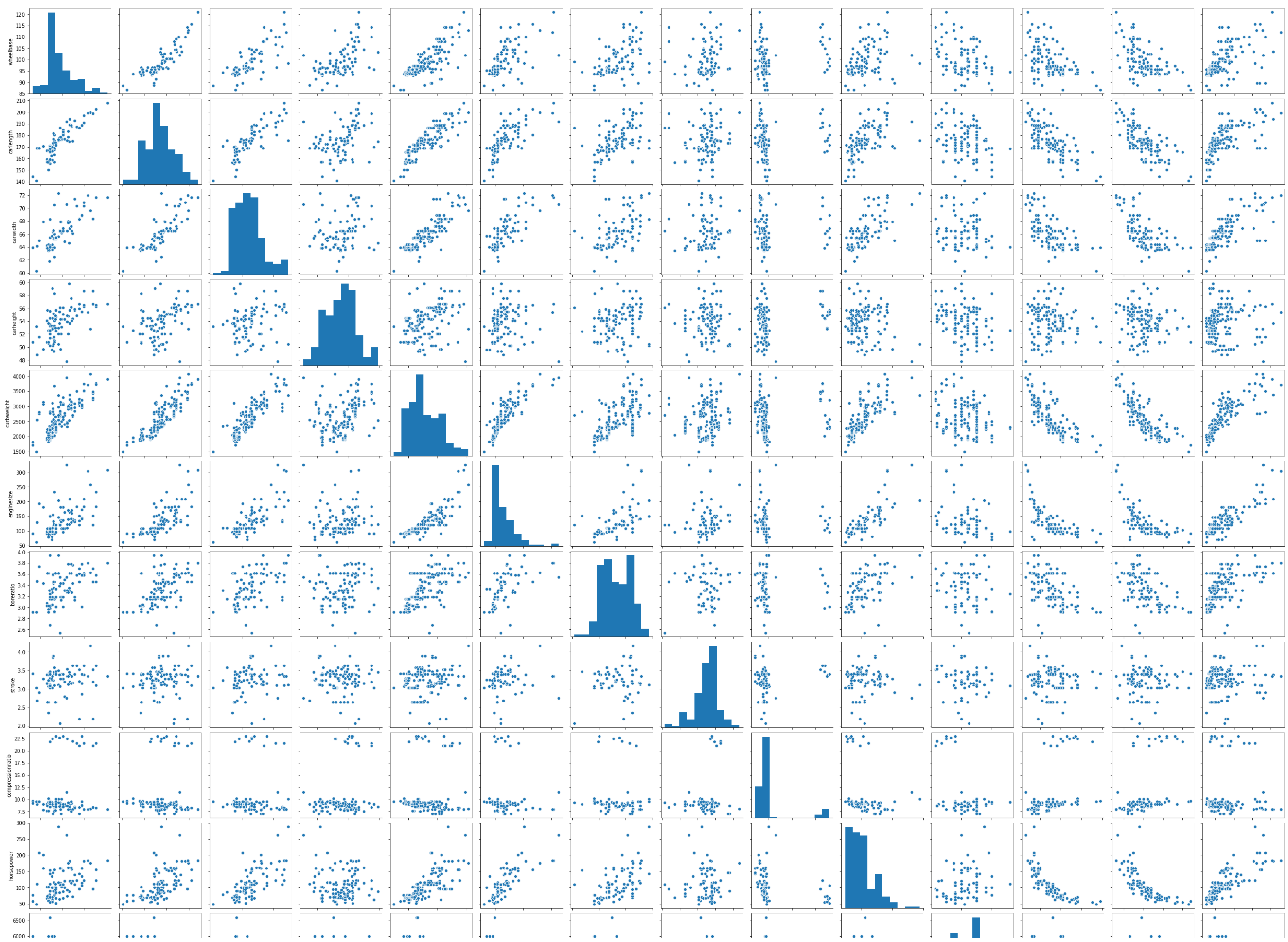
In [26]:

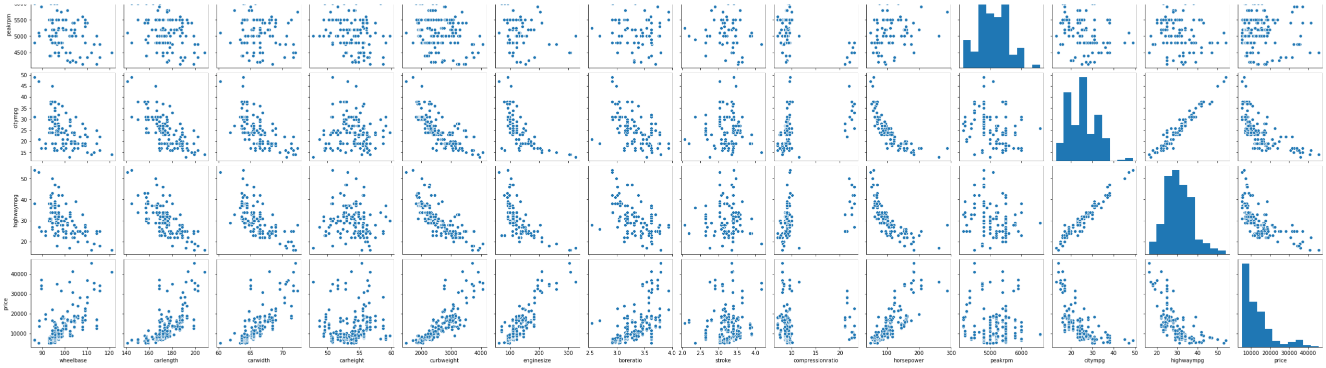
```
######### visualizing the distribution of car prices #########

plt.figure(figsize=(12,10))
plt.title('Car price distribution plot')
sns.distplot(car['price'])
plt.show()   ######### The plots seems to be right skewed, the prices of almost all cars looks like
less than 18000-19000.####
```



In [27]:

```
######## pairplot of all numerical variables ###########

ax= sns.pairplot(car[num_col])
```



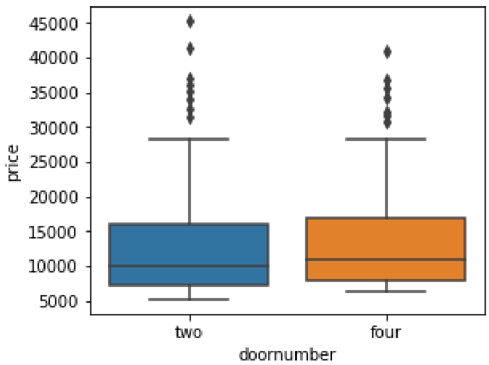


In [ ]:

```
###### Insights:
###carwidth , carlength, curbweight ,enginesize ,horsepowerseems to have a poitive correlation wit
h price.
###carheight doesn't show any significant trend with price.
###citympg , highwaympg - seem to have a significant negative correlation with price. ###########
##
```

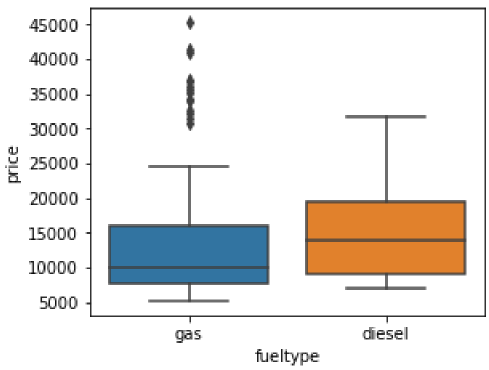
In [28]:

```
########### visualizing of few more chategorical variable #############
##########  univariate and bivariate analysis

plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='doornumber',y='price',data=car)
plt.show()
```



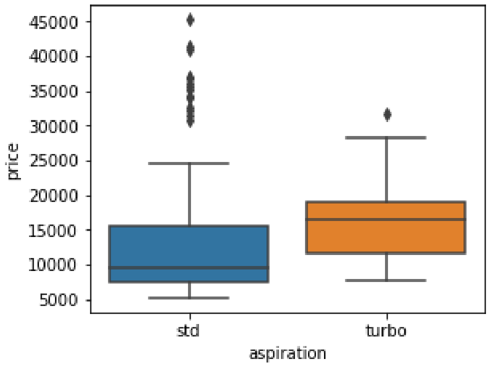
In [29]:

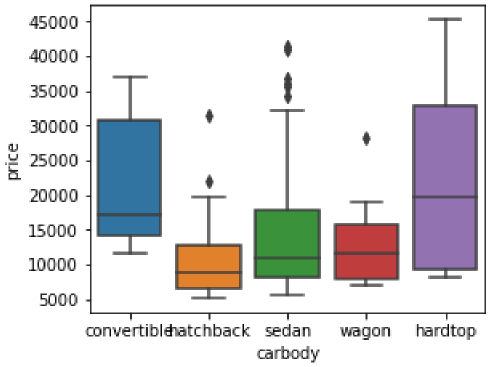
```
plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='fueltype',y='price',data=car)
plt.show()
```

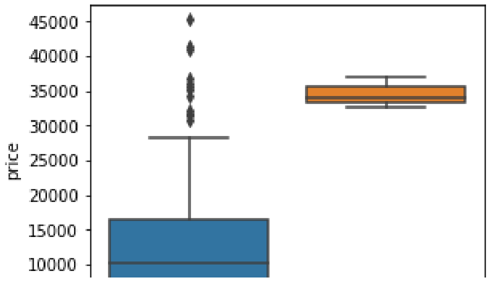


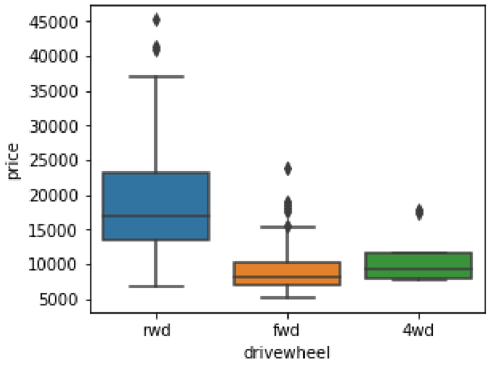
In [30]:

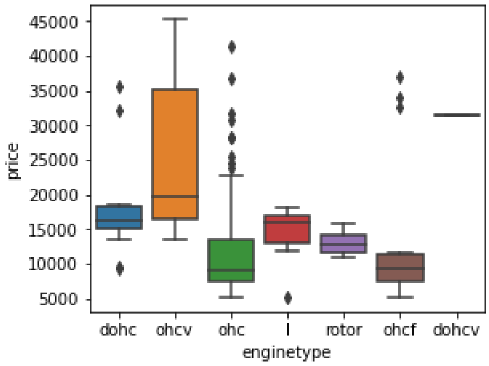
```
plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='aspiration',y='price',data=car)
plt.show()

plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='carbody',y='price',data=car)
plt.show()

plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='enginelocation',y='price',data=car)
plt.show()

plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='drivewheel',y='price',data=car)
plt.show()

plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='enginetype',y='price',data=car)
plt.show()

plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='cylindernumber',y='price',data=car)
plt.show()

plt.figure(figsize=(15,12))
plt.subplot(3,3,1)
sns.boxplot(x='fuelsystem',y='price',data=car)
plt.show()
```

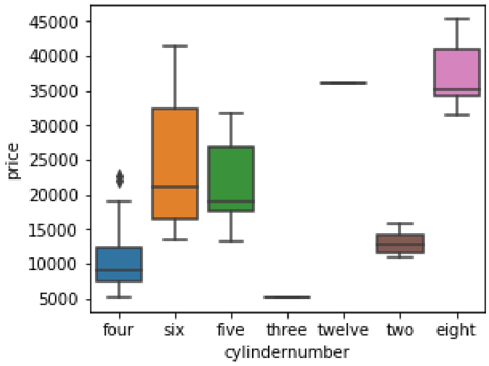


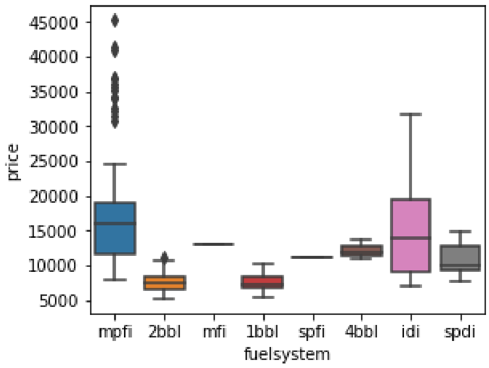












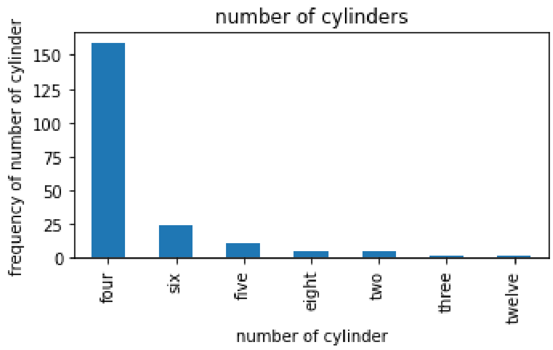
In [ ]:

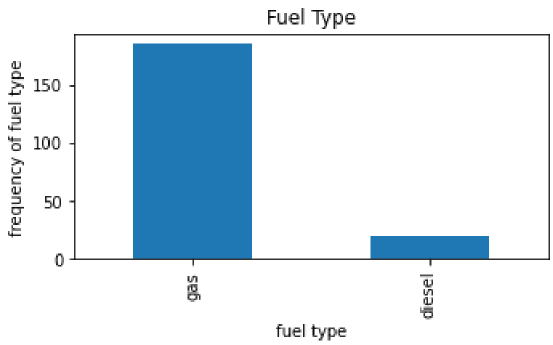
```
#######Insights
#####The cars with fueltype as diesel are comparatively expensive than the cars with fueltype as g
as.
#####All the types of carbody is relatively cheaper as compared to convertible carbody.
#####The cars with rear enginelocation are way expensive than cars with front enginelocation.
#####The price of car is directly proportional to no. of cylinders in most cases.
##### ohcv comes into higher price range cars.
#####DoorNumber isn't affecting the price much
```

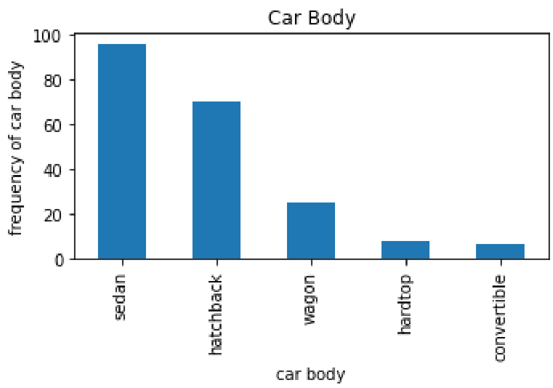
```
#####DoorNumber isn't affecting the price much.
#####HigerEnd cars seems to have rwd drivewheel
```

In [31]:

```
plt.figure(figsize=(18,15))
plt.subplot(5,3,3)
plt1= car['cylindernumber'].value_counts().plot(kind='bar')
plt.title('number of cylinders')
plt1.set(xlabel='number of cylinder',ylabel='frequency of number of cylinder')
plt.show()

plt.figure(figsize=(18,15))
plt.subplot(5,3,3)
plt1= car['fueltype'].value_counts().plot(kind='bar')
plt.title('Fuel Type')
plt1.set(xlabel='fuel type',ylabel='frequency of fuel type')
plt.show()

plt.figure(figsize=(18,15))
plt.subplot(5,3,3)
plt1= car['carbody'].value_counts().plot(kind='bar')
plt.title('Car Body')
plt1.set(xlabel='car body',ylabel='frequency of car body')
plt.show()
```







In [ ]:

```
###Insights:
###The number of cylinders used in most cars is four.
```

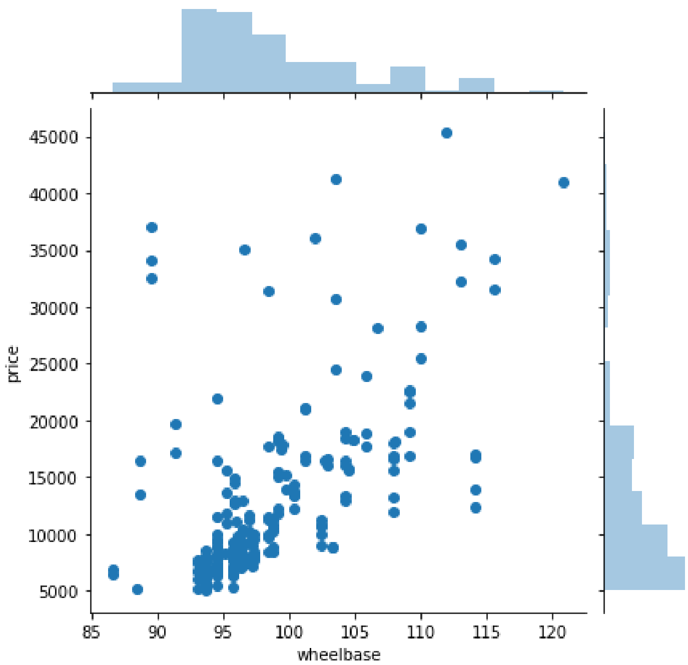
```
###Number of Gas fueled cars are way more than diesel fueled cars.
###Sedan is the most prefered car type.
```

In [32]:

```
sns.jointplot(x='wheelbase',y='price',data=car)
```

Out[32]:

<seaborn.axisgrid.JointGrid at 0x17e2d639c08>

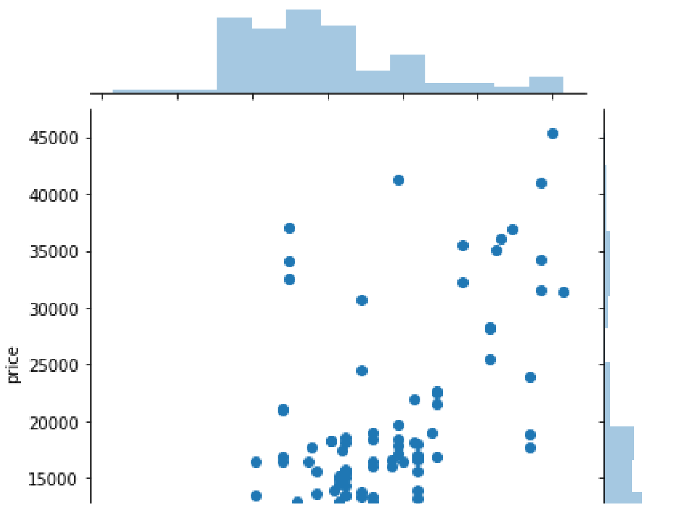


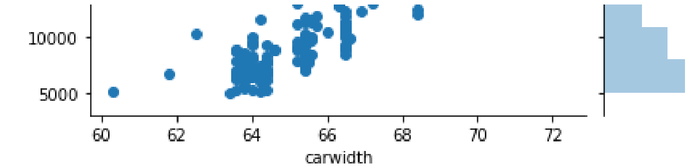
In [33]:

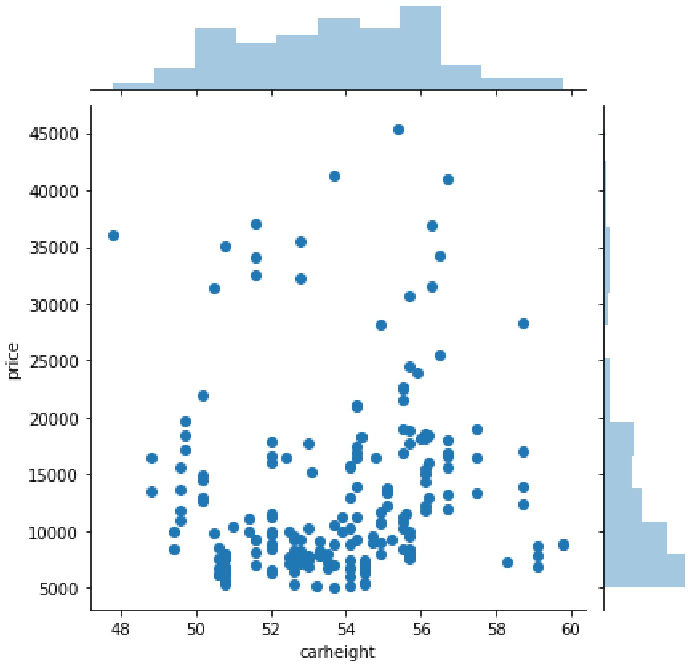
```
sns.jointplot(x='carwidth',y='price',data=car)
sns.jointplot(x='carheight',y='price',data=car)
sns.jointplot(x='curbweight',y='price',data=car)
sns.jointplot(x='enginesize',y='price',data=car)
sns.jointplot(x='boreratio',y='price',data=car)
sns.jointplot(x='stroke',y='price',data=car)
sns.jointplot(x='compressionratio',y='price',data=car)
sns.jointplot(x='horsepower',y='price',data=car)
sns.jointplot(x='peakrpm',y='price',data=car)
sns.jointplot(x='citympg',y='price',data=car)
sns.jointplot(x='highwaympg',y='price',data=car)
```

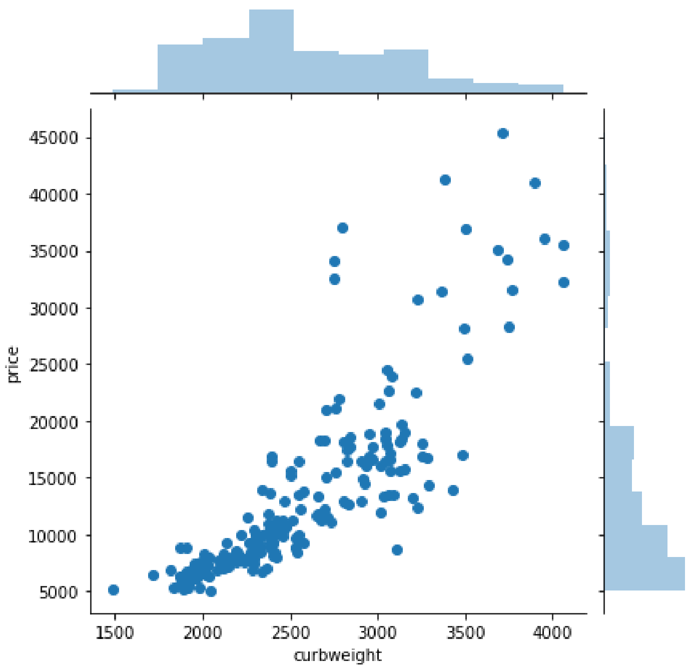
Out[33]:

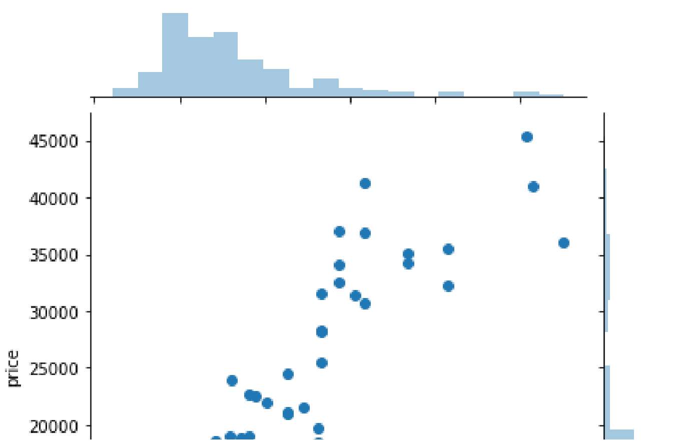
<seaborn.axisgrid.JointGrid at 0x17e2d4550c8>

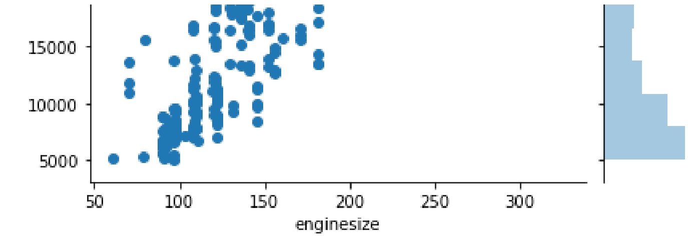


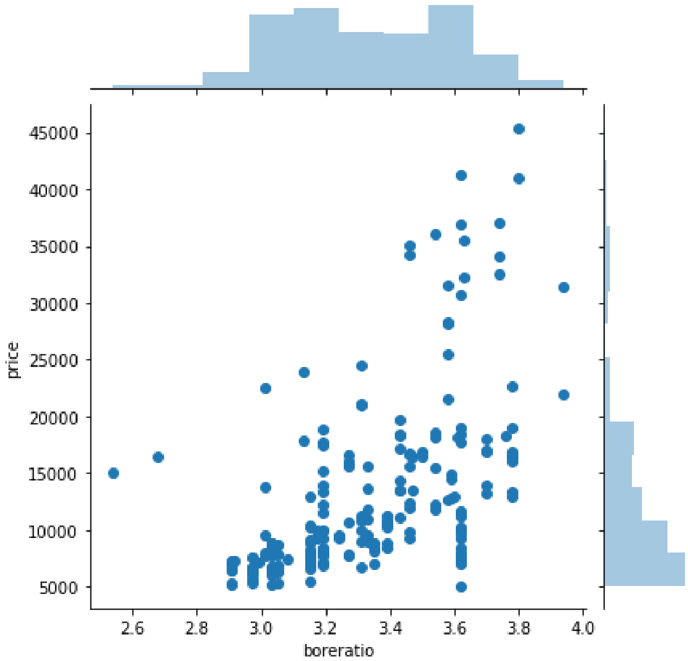

carwidth

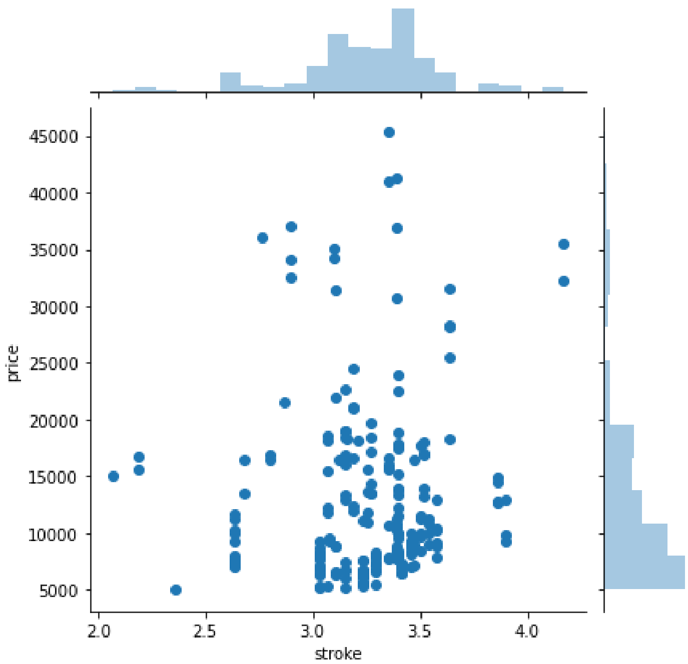

price
carheight

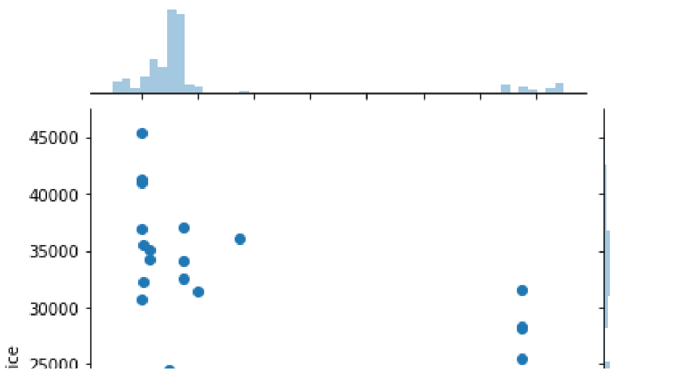

price
curbweight

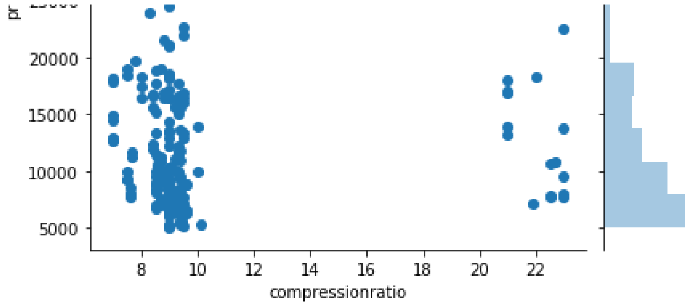

price

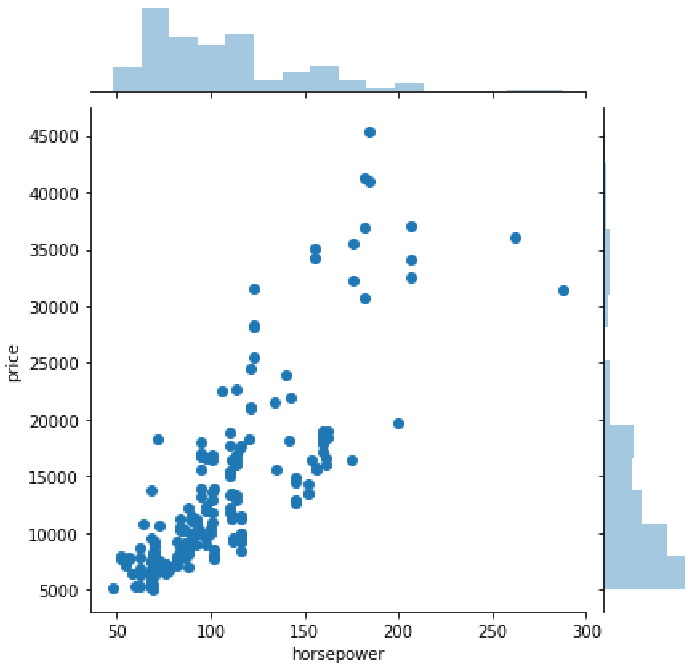

15000
10000
5000
50
100
150
200
250
300
enginesize

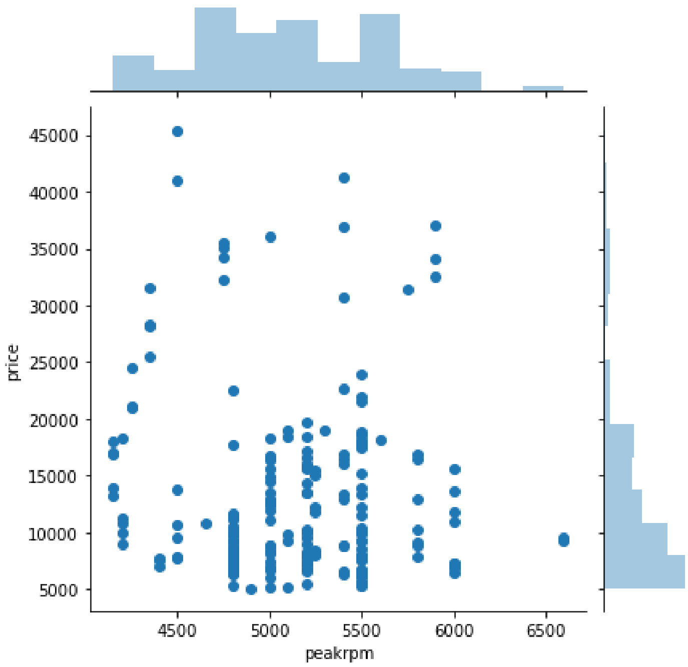

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
boreratio

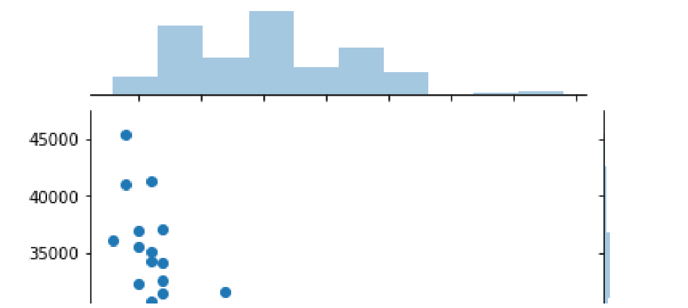

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
2.0
2.5
3.0
3.5
4.0
stroke

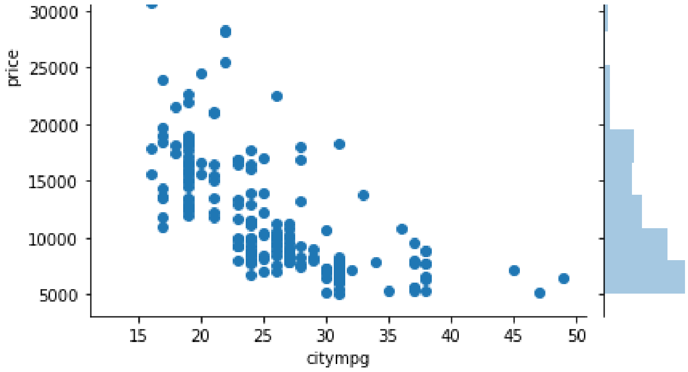

45000
40000
35000
30000
25000

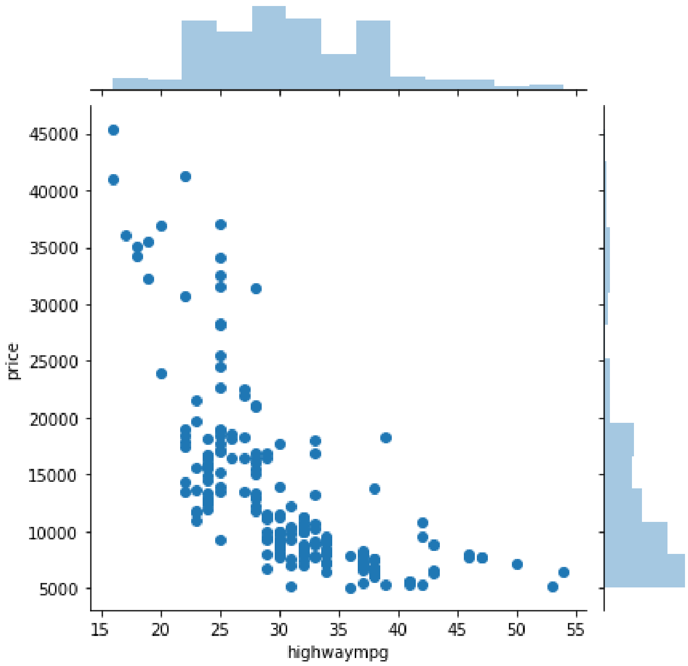

20000
15000
10000
5000
8
10
12
14
16
18
20
22
compressionratio


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
50
100
150
200
250
300
horsepower


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
4500
5000
5500
6000
6500
peakrpm

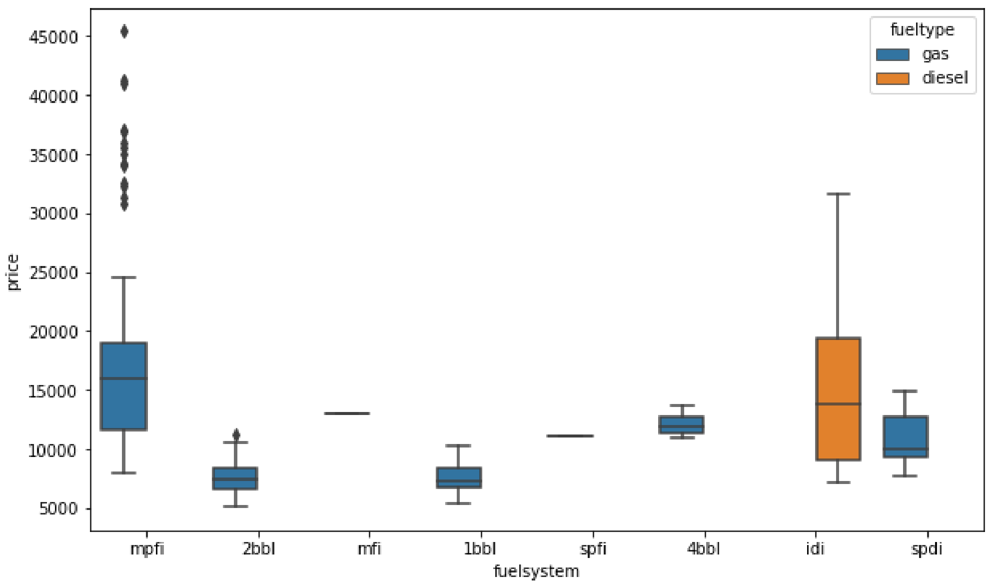

45000
40000
35000





In [34]:

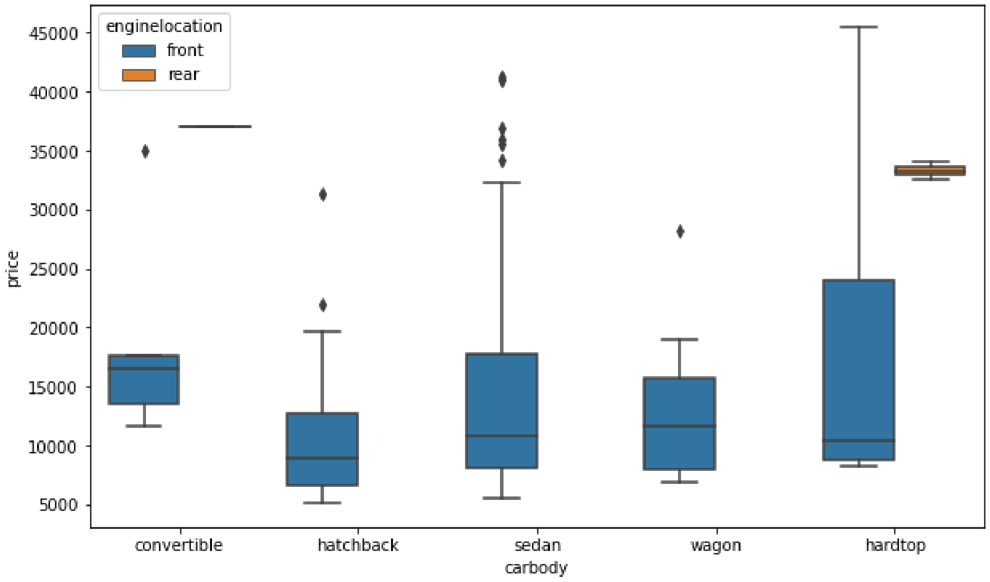
```
plt.figure(figsize = (10, 6))
sns.boxplot(x = 'fuelsystem', y = 'price', hue = 'fueltype', data = car)
plt.show()
```



In [35]:

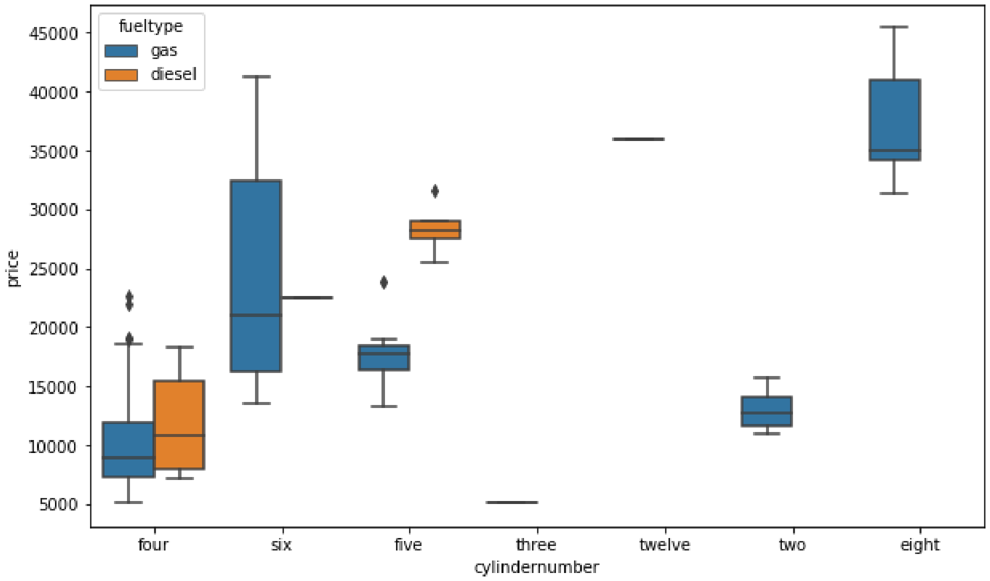
```
plt.figure(figsize = (10, 6))
sns.boxplot(x = 'carbody', y = 'price', hue = 'enginelocation', data = car)
plt.show()
```

```
plt.show()
```



In [36]:

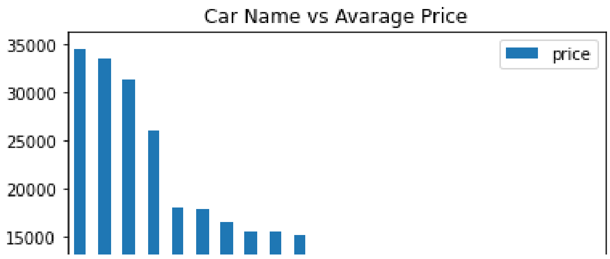
```
plt.figure(figsize = (10, 6))
sns.boxplot(x = 'cylindernumber', y = 'price', hue = 'fueltype', data = car)
plt.show()
```

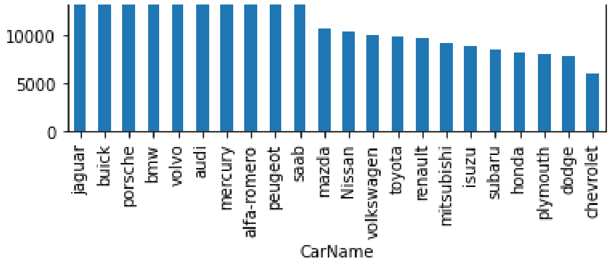


In [37]:

```
plt.figure(figsize=(20,15))
car_bar= pd.DataFrame(car.groupby(['CarName'])['price'].mean().sort_values(ascending=False))
car_bar.plot.bar()
plt.title('Car Name vs Avarage Price')
plt.show()
```

<Figure size 1440x1080 with 0 Axes>





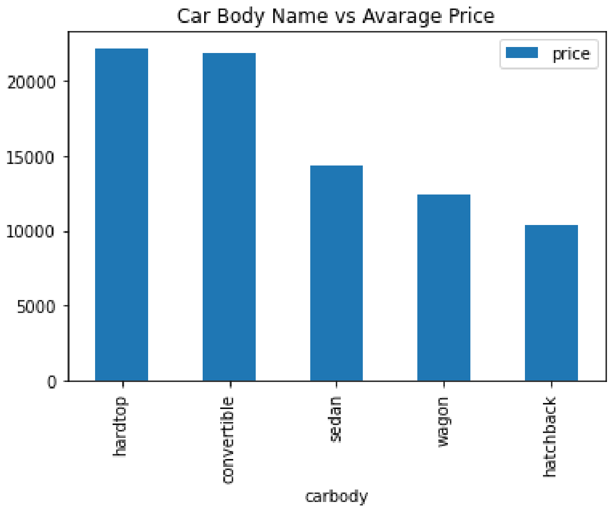
In [ ]:

```
#####Insights:
#####Jaguar,Buick and porsche seems to have the highest average price.
```

In [38]:

```
plt.figure(figsize=(20,18))
car_body= pd.DataFrame(car.groupby(['carbody'])['price'].mean().sort_values(ascending=False))
car_body.plot.bar()
plt.title('Car Body Name vs Avarage Price')
plt.show()
```

```
<Figure size 1440x1296 with 0 Axes>
```

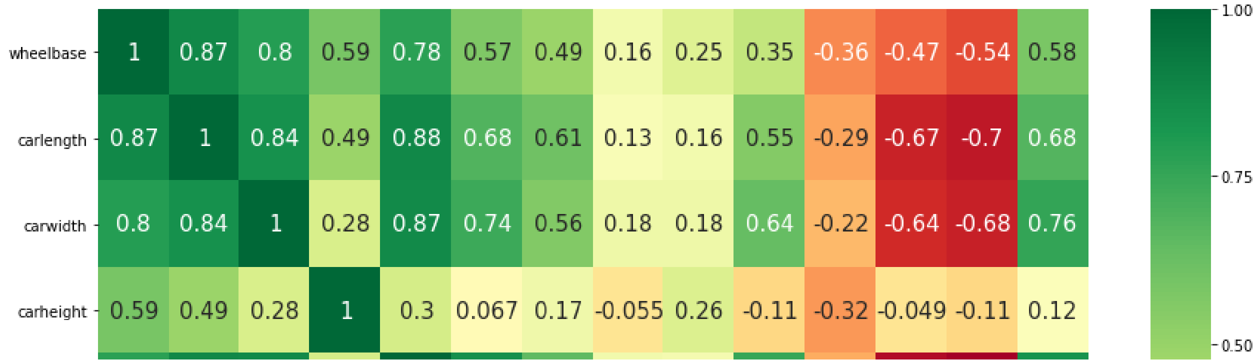


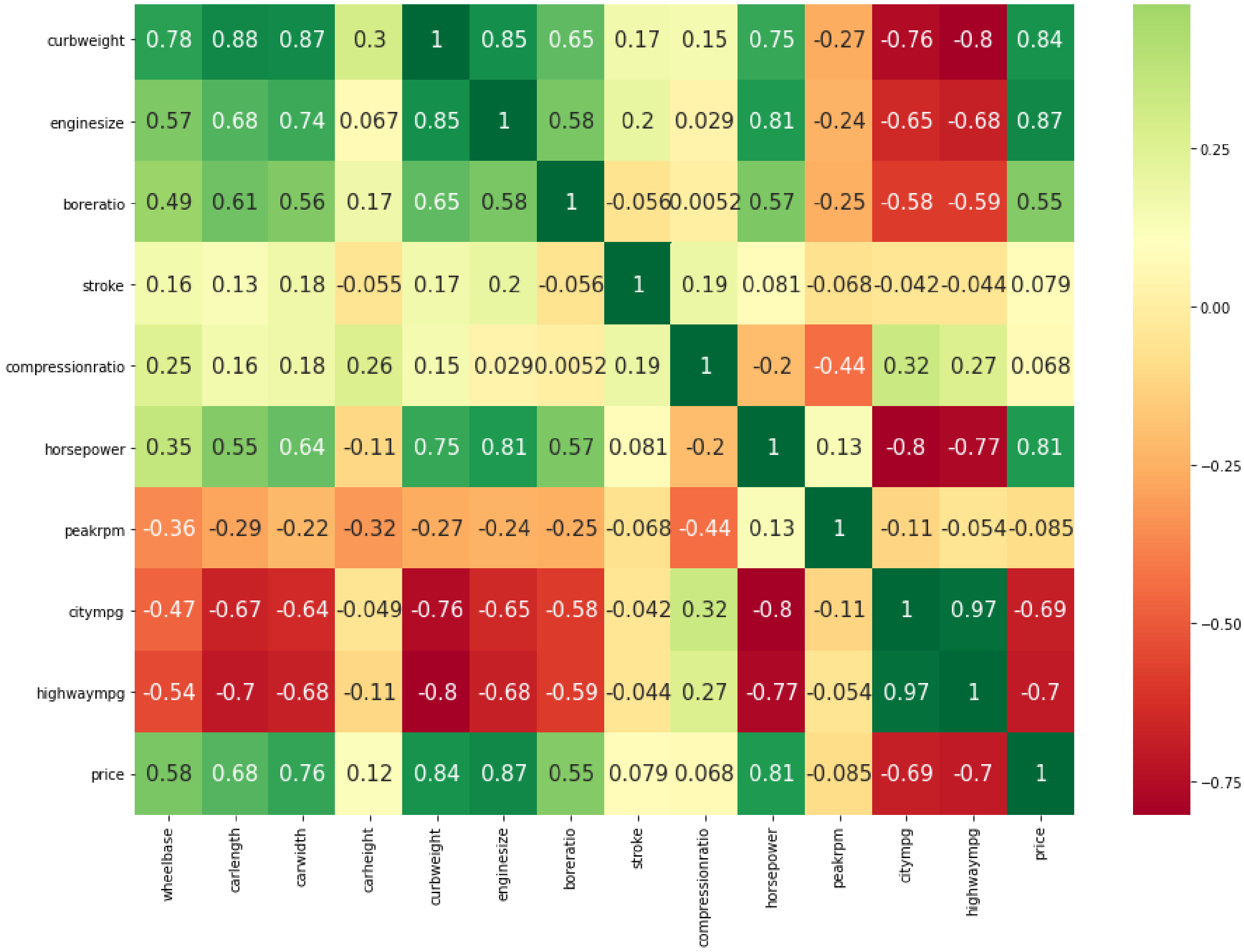
In [ ]:

```
#####Insights:
#####hardtop and convertible seems to have the highest average price.
```

In [39]:

```
######## finding correlation using heatmap #########
plt.figure(figsize=(15,15))
sns.heatmap(car.corr(),annot=True,cmap="RdYlGn",annot_kws={"size":15})
plt.show()
```





In [ ]:

```
####### insights
####### Horsepower,boreratio,enginesize,curbweight,carlength,carweight,carwidth,wheelbase has corr
elation with price
####### Highwaymap,citymap has negative correlation with price
####### peakrpm,stroke,compressionratio,carheight has low correlation with price
```

In [40]:

```
#Binning the Car Companies based on avg prices of each car Company.

car['price'] = car['price'].astype('int')
car_temp = car.copy()
t = car_temp.groupby(['CarName'])['price'].mean()
print(t)
car_temp = car_temp.merge(t.reset_index(), how='left', on='CarName')
bins = [0,10000,20000,40000]
label =['Budget_Friendly','Medium_Range','TopNotch_Cars']
car['Cars_Category'] = pd.cut(car_temp['price_y'], bins, right=False, labels=label)
car.head(20)
```

```
CarName
Nissan         10415.666667
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
peugeot        15489.090909
```

```
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
```

Out[40]:

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	carlength	...	fuelsyste
0	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	...	mp
1	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	...	mp
2	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	...	mp
3	2	audi	gas	std	four	sedan	fwd	front	99.8	176.6	...	mp
4	2	audi	gas	std	four	sedan	4wd	front	99.4	176.6	...	mp
5	2	audi	gas	std	two	sedan	fwd	front	99.8	177.3	...	mp
6	1	audi	gas	std	four	sedan	fwd	front	105.8	192.7	...	mp
7	1	audi	gas	std	four	wagon	fwd	front	105.8	192.7	...	mp
8	1	audi	gas	turbo	four	sedan	fwd	front	105.8	192.7	...	mp
9	0	audi	gas	turbo	two	hatchback	4wd	front	99.5	178.2	...	mp
10	2	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	...	mp
11	0	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	...	mp
12	0	bmw	gas	std	two	sedan	rwd	front	101.2	176.8	...	mp
13	0	bmw	gas	std	four	sedan	rwd	front	101.2	176.8	...	mp
14	1	bmw	gas	std	four	sedan	rwd	front	103.5	189.0	...	mp
15	0	bmw	gas	std	four	sedan	rwd	front	103.5	189.0	...	mp
16	0	bmw	gas	std	two	sedan	rwd	front	103.5	193.8	...	mp
17	0	bmw	gas	std	four	sedan	rwd	front	110.0	197.0	...	mp
18	2	chevrolet	gas	std	two	hatchback	fwd	front	88.4	141.1	...	2b
19	1	chevrolet	gas	std	two	hatchback	fwd	front	94.5	155.9	...	2b

20 rows × 26 columns

In [41]:

```
sig_col = ['price','Cars_Category','enginetype','fueltype',
'aspiration','carbody','cylindernumber', 'drivewheel',
            'wheelbase','curbweight', 'enginesize', 'boreratio','horsepower',
                    'citympg','highwaympg', 'carlength','carwidth']
```

In [42]:

```
car= car[sig_col]
```

In [44]:

```
####### data preparation, dummy variable #########

####The variable carbody has five levels. We need to convert these levels into integer.
####Similarly we need to convert the categorical variables to numeric.

sig_cat_col =
['Cars_Category','fueltype','aspiration','carbody','drivewheel','enginetype','cylindernumber']
```

In [45]:

```
dummies = pd.get_dummies(car[sig_cat_col])
```

In [46]:

```
dummies.shape
```

Out[46]:

```
(205, 29)
```

In [47]:

```
dummies = pd.get_dummies(car[sig_cat_col], drop_first = True)
dummies.shape
```

Out[47]:

```
(205, 22)
```

In [48]:

```
# Add the results to the original dataframe


car = pd.concat([car, dummies], axis = 1)
```

In [49]:

```
# Drop the original cat variables as dummies are already created

car.drop( sig_cat_col, axis = 1, inplace = True)
car.shape
```

Out[49]:

```
(205, 32)
```

In [50]:

```
car
```

Out[50]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	citympg	highwaympg	carlength	carwidth	...	enginetype_ohc
0	13495	88.6	2548	130	3.47	111	21	27	168.8	64.1	...	0
1	16500	88.6	2548	130	3.47	111	21	27	168.8	64.1	...	0
2	16500	94.5	2823	152	2.68	154	19	26	171.2	65.5	...	0
3	13950	99.8	2337	109	3.19	102	24	30	176.6	66.2	...	1
4	17450	99.4	2824	136	3.19	115	18	22	176.6	66.4	...	1
...	...	...	...	...	...	...	...	...	...	...	...	...
200	16845	109.1	2952	141	3.78	114	23	28	188.8	68.9	...	1
201	19045	109.1	3049	141	3.78	160	19	25	188.8	68.8	...	1
202	21485	109.1	3012	173	3.58	134	18	23	188.8	68.9	...	0
203	22470	109.1	3217	145	3.01	106	26	27	188.8	68.9	...	1
204	22625	109.1	3062	141	3.78	114	19	25	188.8	68.9	...	1

205 rows × 32 columns

In [52]:

```
######## splitting the data into train and test ###########
```

```
df_train,df_test= train_test_split(car,train_size=0.8,test_size=0.2,random_state=100)
```

In [54]:

```
df_train.head()
```

Out[54]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	citympg	highwaympg	carlength	carwidth	...	enginetype_ohc
3	13950	99.8	2337	109	3.19	102	24	30	176.6	66.2	...	1
157	7198	95.7	2109	98	3.19	70	30	37	166.3	64.4	...	1
81	8499	96.3	2328	122	3.35	88	25	32	173.0	65.4	...	1
32	5399	93.7	1837	79	2.91	60	38	42	150.0	64.0	...	1
99	8949	97.2	2324	120	3.33	97	27	34	173.4	65.2	...	1

5 rows × 32 columns

In [ ]:

```
######Rescaling the Features
######For Simple Linear Regression, scaling doesn't impact model. So it is extremely important to rescale the variables so that they have a comparable scale. If we don't have comparable scales, then some of the coefficients as obtained by fitting the regression model might be very large or very small as compared to the other coefficients. There are two common ways of rescaling:

######Min-Max scaling
######Standardisation (mean-0, sigma-1)
```

In [55]:

```
scaler = preprocessing.StandardScaler()
sig_num_col = ['wheelbase','carlength','carwidth','curbweight','enginesize','boreratio','horsepower','citympg','highwaympg','price']
```

In [56]:

```
###### applying scaler function to all except the dummy variable #############

import warnings
warnings.filterwarnings("ignore")
df_train[sig_num_col]=scaler.fit_transform(df_train[sig_num_col])
```

In [57]:

```
df_train.head()
```

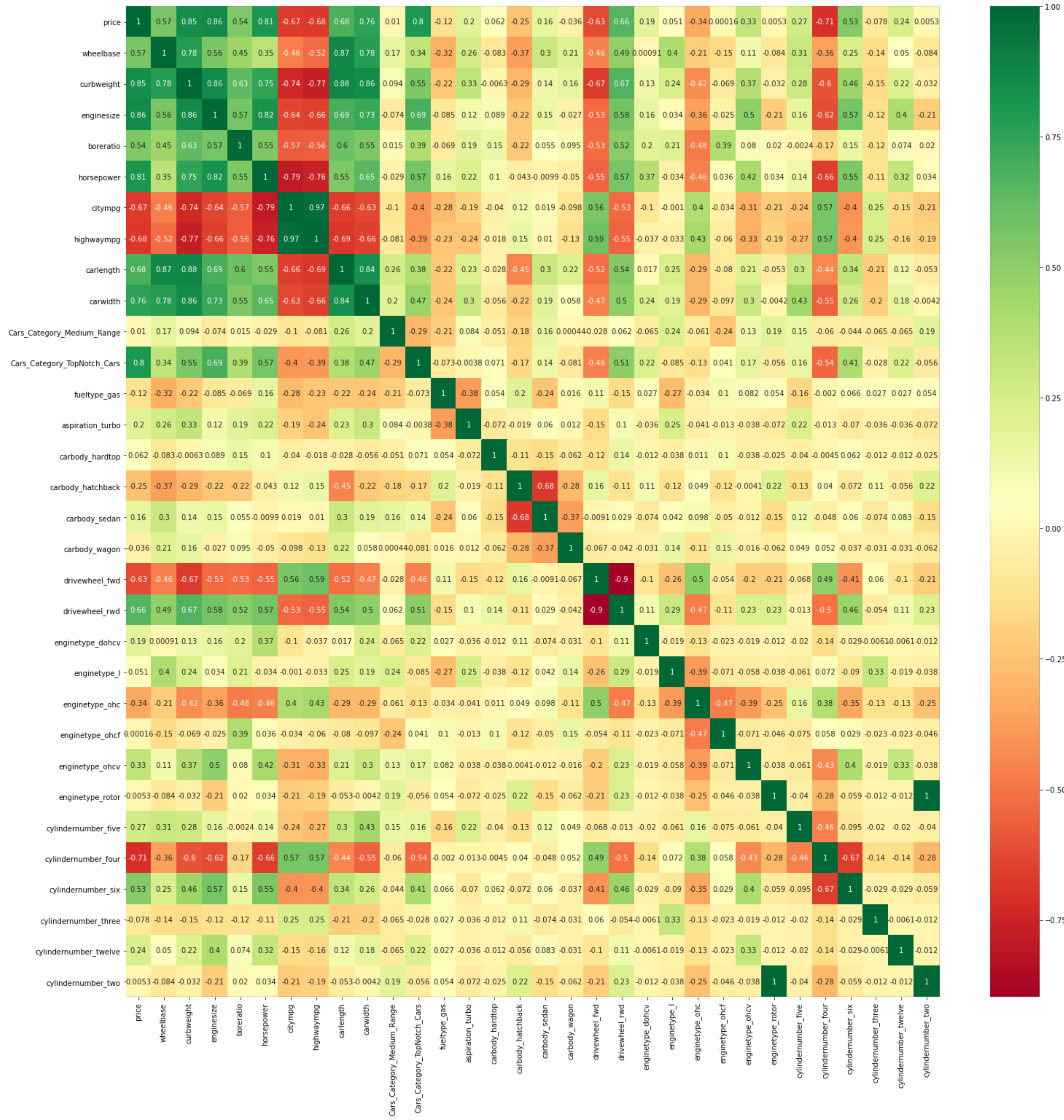
Out[57]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	citympg	highwaympg	carlength	carwidth	...	enginetype_
3	0.155048	0.256524	-0.343330	-0.372828	-0.419206	0.024240	-0.254620	-0.183655	0.290980	0.205880	...	
157	-0.725489	-0.460676	-0.781780	-0.652371	-0.419206	-0.791203	0.646625	0.825130	-0.558965	-0.630100	...	
81	-0.555824	-0.355720	-0.360637	-0.042458	0.205437	-0.332517	-0.104413	0.104569	-0.006088	-0.165667	...	
32	-0.960099	-0.810530	-1.304842	-1.135218	-1.512333	-1.046029	1.848286	1.545690	-1.904025	-0.815873	...	
99	-0.497139	-0.198286	-0.368329	-0.093284	0.127357	-0.103173	0.196003	0.392793	0.026919	-0.258553	...	

5 rows × 32 columns

In [59]:

```
# Let's check the correlation coefficients to see which variables are highly correlated

plt.figure(figsize = (25, 25))
sns.heatmap(df_train.corr(),annot=True, cmap="RdYlGn")
plt.show() ####### we can see
curbweight,enginesize,horsepower,carlength,carwidth,cars_category_topnotch_cars,driverwheel_rwd,cy
r number six, wheel base..these are some highly correlated with price##########
```



In [60]:

```
######## dividing into X and y set for model building ###########

X_train=df_train
y_train=df_train.pop('price')
```

In [61]:

```
########## building a linear model ########

X_train_1=X_train['horsepower']
```

In [62]:

```
####### adding a constant #########
X_train_1c= sm.add_constant(X_train_1)##### It's because you expect your dependent variable to tak
e a nonzero value when all the otherwise included regressors are set to zero
```

In [63]:

```
##### create a first fitted model########
lr_1=sm.OLS(y_train,X_train_1c).fit()
```

In [64]:

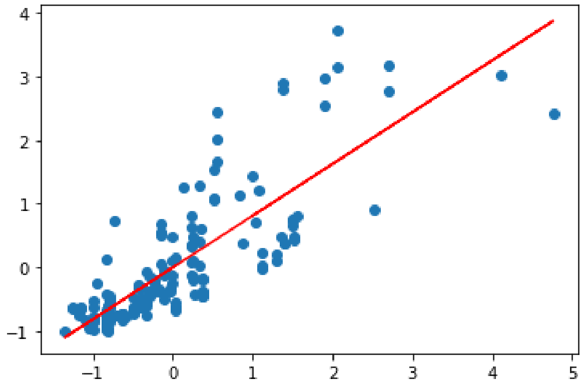
```
lr_1.params
```

Out[64]:

```
const         6.938894e-17
horsepower    8.126066e-01
dtype: float64
```

In [66]:

```
plt.scatter(X_train_1c.iloc[:,1],y_train)
plt.plot(X_train_1c.iloc[:,1],0.8126*X_train_1c.iloc[:,1],'r')
plt.show() ##### the correlation coefficient r measures the strength and direction of a linear rel
ationship between two variables on a scatterplot.
################The value of r is always between +1 and -1.
```



In [67]:

```
print(lr_1.summary())  ###### R squared and Adj.R squared value is 0.66 and 0.65##########
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.660
Model:                            OLS   Adj. R-squared:                  0.658
Method:                 Least Squares   F-statistic:                     314.9
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           7.96e-40
Time:                        00:36:55   Log-Likelihood:                -144.16
No. Observations:                 164   AIC:                             292.3
Df Residuals:                     162   BIC:                             298.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       6.939e-17      0.046   1.52e-15      1.000      -0.090       0.090
horsepower     0.8126      0.046     17.746      0.000       0.722       0.903
==============================================================================
Omnibus:                       36.811   Durbin-Watson:                   1.855
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               61.090
Skew:                           1.126   Prob(JB):                     5.42e-14
Kurtosis:                       4.967   Cond. No.                         1.00
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [68]:

```
######## adding another variable ###########

X_train_2 = X_train[['horsepower', 'curbweight']]
```

In [69]:

```
# Add a constant
X_train_2c = sm.add_constant(X_train_2)

# Create a second fitted model
lr_2 = sm.OLS(y_train, X_train_2c).fit()
```

In [70]:

```
lr_2.params
```

Out[70]:

```
const         6.938894e-17
horsepower    4.066668e-01
curbweight    5.391626e-01
dtype: float64
```

In [71]:

```
print(lr_2.summary())#### R squared and Adjusted.R squared is 0.78, R-squared has been increased.
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.786
Model:                            OLS   Adj. R-squared:                  0.784
Method:                 Least Squares   F-statistic:                     296.1
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           1.15e-54
Time:                        00:39:37   Log-Likelihood:                -106.19
No. Observations:                 164   AIC:                             218.4
Df Residuals:                     161   BIC:                             227.7
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       6.939e-17      0.036    1.9e-15      1.000      -0.072       0.072
horsepower     0.4067      0.055      7.345      0.000       0.297       0.516
curbweight     0.5392      0.055      9.738      0.000       0.430       0.648
==============================================================================
Omnibus:                       37.859   Durbin-Watson:                   1.788
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               87.458
Skew:                           0.987   Prob(JB):                     1.02e-19
Kurtosis:                       5.983   Cond. No.                         2.66
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [72]:

```
######## adding another variable#######
X_train_3 = X_train[['horsepower', 'curbweight', 'enginesize']]
```

In [73]:

```
# Add a constant
X_train_3c = sm.add_constant(X_train_3)

# Create a third fitted model
lr_3 = sm.OLS(y_train, X_train_3c).fit()
```

```
lr_3 = sm.OLS(y_train, X_train_3c).fit()
lr_3.params
```

Out[73]:

```
const         6.938894e-17
horsepower    2.721143e-01
curbweight    3.484593e-01
enginesize    3.379127e-01
dtype: float64
```

In [74]:

```
print(lr_3.summary()) ### R squared and Adjusted R squared has been increased to 0.80#########
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.807
Model:                            OLS   Adj. R-squared:                  0.804
Method:                 Least Squares   F-statistic:                     223.5
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           5.60e-57
Time:                        00:41:37   Log-Likelihood:                -97.678
No. Observations:                 164   AIC:                             203.4
Df Residuals:                     160   BIC:                             215.8
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       6.939e-17      0.035      2e-15      1.000      -0.069       0.069
horsepower     0.2721      0.062      4.406      0.000       0.150       0.394
curbweight     0.3485      0.070      4.999      0.000       0.211       0.486
enginesize     0.3379      0.081      4.183      0.000       0.178       0.497
==============================================================================
Omnibus:                       29.472   Durbin-Watson:                   1.843
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               65.487
Skew:                           0.778   Prob(JB):                     6.02e-15
Kurtosis:                       5.676   Cond. No.                         4.66
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
##### We have achieved a R-squared of 0.819 by manually picking the highly correlated variables fr
om heatmap
##### Now lets use RFE to select the independent variables which accurately predicts the dependent
variable price.
##### RFE
##### Let's use Recursive feature elimination since we have too many independent variables
```

In [75]:

```
####  Running RFE with the output number of the variable equal to 15 ####
lm=LinearRegression()
lm.fit(X_train,y_train)
rfe=RFE(lm,15)
rfe= rfe.fit(X_train,y_train)
```

In [76]:

```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[76]:

```
[('wheelbase', False, 9),
 ('curbweight', True, 1),
 ('enginesize', False, 12),
 ('boreratio', False, 8),
 ('horsepower', True, 1),
 ('citympg', False, 5),
```

```
 ('highwaympg', False, 10),
 ('carlength', False, 13),
 ('carwidth', True, 1),
 ('Cars_Category_Medium_Range', False, 2),
 ('Cars_Category_TopNotch_Cars', True, 1),
 ('fueltype_gas', False, 14),
 ('aspiration_turbo', False, 15),
 ('carbody_hardtop', True, 1),
 ('carbody_hatchback', True, 1),
 ('carbody_sedan', True, 1),
 ('carbody_wagon', True, 1),
 ('drivewheel_fwd', False, 11),
 ('drivewheel_rwd', False, 6),
 ('enginetype_dohcv', True, 1),
 ('enginetype_l', True, 1),
 ('enginetype_ohc', True, 1),
 ('enginetype_ohcf', True, 1),
 ('enginetype_ohcv', True, 1),
 ('enginetype_rotor', False, 17),
 ('cylindernumber_five', True, 1),
 ('cylindernumber_four', True, 1),
 ('cylindernumber_six', False, 4),
 ('cylindernumber_three', False, 7),
 ('cylindernumber_twelve', False, 3),
 ('cylindernumber_two', False, 16)]
```

In [77]:

```
# Selecting the variables which are in support
col_imp=X_train.columns[rfe.support_]
col_imp
```

Out[77]:

```
Index(['curbweight', 'horsepower', 'carwidth', 'Cars_Category_TopNotch_Cars',
       'carbody_hardtop', 'carbody_hatchback', 'carbody_sedan',
       'carbody_wagon', 'enginetype_dohcv', 'enginetype_l', 'enginetype_ohc',
       'enginetype_ohcf', 'enginetype_ohcv', 'cylindernumber_five',
       'cylindernumber_four'],
      dtype='object')
```

In [78]:

```
# Creating X_train dataframe with RFE selected variables
X_train_rfe=X_train[col_imp]
X_train_rfe
```

Out[78]:

	curbweight	horsepower	carwidth	Cars_Category_TopNotch_Cars	carbody_hardtop	carbody_hatchback	carbody_sedan	carbody_
3	-0.343330	0.024240	0.205880	0	0	0	1	
157	-0.781780	-0.791203	-0.630100	0	0	1	0	
81	-0.360637	-0.332517	-0.165667	0	0	1	0	
32	-1.304842	-1.046029	-0.815873	0	0	1	0	
99	-0.368329	-0.103173	-0.258553	0	0	1	0	
...	...	...	...	...	...	...	...	
87	-0.216411	0.380996	-0.165667	0	0	0	1	
103	1.047016	1.298369	0.345210	0	0	0	1	
67	1.921992	0.559374	2.110055	1	0	0	1	
24	-1.054849	-0.842168	-0.908759	0	0	1	0	
8	1.097015	0.992578	2.620931	0	0	0	1	

164 rows × 15 columns

In [ ]:

```
###### After passing the arbitary selected columns by RFE we will manually evaluate each models p-
value and VIF value. Unless we find the acceptable range for p-values and VIF we keep dropping the
variables one at a time based on below criteria.

##### High p-value High VIF : Drop the variable
##### High p-value Low VIF or Low p-value High VIF : Drop the variable with high p-value first
##### Low p-value Low VIF : accept the variable
```

In [79]:

```
# Adding a constant variable and Build a first fitted model

import statsmodels.api as sm
X_train_rfec=sm.add_constant(X_train_rfe)
lm_rfe= sm.OLS(y_train,X_train_rfec).fit()

#Summary of linear model
print(lm_rfe.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.938
Model:                            OLS   Adj. R-squared:                  0.932
Method:                 Least Squares   F-statistic:                     149.6
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           2.59e-81
Time:                        01:01:17   Log-Likelihood:                -4.5365
No. Observations:                 164   AIC:                             41.07
Df Residuals:                     148   BIC:                             90.67
Df Model:                          15
Covariance Type:            nonrobust
===============================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
const                           0.8039      0.150      5.353      0.000       0.507       1.101
curbweight                      0.2256      0.062      3.646      0.000       0.103       0.348
horsepower                      0.2570      0.045      5.694      0.000       0.168       0.346
carwidth                        0.2197      0.048      4.586      0.000       0.125       0.314
Cars_Category_TopNotch_Cars     1.0507      0.101     10.381      0.000       0.851       1.251
carbody_hardtop                -0.6596      0.184     -3.581      0.000      -1.024      -0.296
carbody_hatchback              -0.8164      0.133     -6.150      0.000      -1.079      -0.554
carbody_sedan                  -0.7020      0.132     -5.314      0.000      -0.963      -0.441
carbody_wagon                  -0.8287      0.143     -5.796      0.000      -1.111      -0.546
enginetype_dohcv               -0.8684      0.327     -2.652      0.009      -1.515      -0.221
enginetype_l                    0.2664      0.132      2.015      0.046       0.005       0.528
enginetype_ohc                  0.3051      0.107      2.860      0.005       0.094       0.516
enginetype_ohcf                 0.2761      0.125      2.209      0.029       0.029       0.523
enginetype_ohcv                -0.1792      0.117     -1.533      0.127      -0.410       0.052
cylindernumber_five            -0.3776      0.135     -2.792      0.006      -0.645      -0.110
cylindernumber_four            -0.5267      0.100     -5.271      0.000      -0.724      -0.329
==============================================================================
Omnibus:                       35.523   Durbin-Watson:                   2.154
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               84.840
Skew:                           0.911   Prob(JB):                     3.78e-19
Kurtosis:                       6.016   Cond. No.                         28.6
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
############ Looking at the p-values, it looks like some of the variables aren't really significan
t (in the presence of other variables)
#####and we need to drop it
##### Variance Inflation Factor or VIF, gives a basic quantitative idea about how much the feature
variables are correlated with each other.
#### It is an extremely important parameter to test our linear model. The formula for calculating
VIF is:

VIFi=1/1-Ri^2 ############
```

In [81]:

```
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif=pd.DataFrame()
vif['Features']=X_train_rfe.columns
vif['VIF']=[variance_inflation_factor(X_train_rfe.values,i) for i in range(X_train_rfe.shape[1])]
vif['VIF']=round(vif['VIF'],2)
vif=vif.sort_values(by='VIF',ascending=False)
vif
```

Out[81]:

	Features	VIF
10	enginetype_ohc	19.56
14	cylindernumber_four	14.58
0	curbweight	9.12
6	carbody_sedan	7.20
2	carwidth	5.25
5	carbody_hatchback	4.88
1	horsepower	4.61
7	carbody_wagon	3.14
11	enginetype_ohcf	2.95
3	Cars_Category_TopNotch_Cars	2.49
13	cylindernumber_five	2.43
9	enginetype_l	2.28
12	enginetype_ohcv	1.63
8	enginetype_dohcv	1.56
4	carbody_hardtop	1.45

In [ ]:

```
###### We generally want a VIF that is less than 5. So there are clearly some variables we need to drop.
##### Dropping the variable and updating the model
##### Dropping enginetype_ohcv beacuse its p-value is 0.393 and we want p-value less than 0.05 and hence rebuilding the model
```

In [82]:

```
X_train_rfe1 = X_train_rfe.drop('enginetype_ohcv', 1,)

# Adding a constant variable and Build a second fitted model

X_train_rfe1c = sm.add_constant(X_train_rfe1)
lm_rfe1 = sm.OLS(y_train, X_train_rfe1c).fit()

#Summary of linear model
print(lm_rfe1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.937
Model:                            OLS   Adj. R-squared:                  0.931
Method:                 Least Squares   F-statistic:                     158.7
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           6.42e-82
Time:                        01:22:01   Log-Likelihood:                -5.8277
No. Observations:                 164   AIC:                             41.66
Df Residuals:                     149   BIC:                             88.15
Df Model:                          14
Covariance Type:            nonrobust
===============================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------------------------
const                          0.7350      0.144      5.107      0.000       0.451       1.019
curbweight                     0.2292      0.062      3.691      0.000       0.107       0.352
horsepower                     0.2523      0.045      5.577      0.000       0.163       0.342
carwidth                       0.2100      0.048      4.402      0.000       0.116       0.304
Cars_Category_TopNotch_Cars    1.0601      0.101     10.447      0.000       0.860       1.261
carbody_hardtop               -0.6529      0.185     -3.530      0.001      -1.018      -0.287
carbody_hatchback             -0.8131      0.133     -6.099      0.000      -1.077      -0.550
carbody_sedan                 -0.7010      0.133     -5.283      0.000      -0.963      -0.439
carbody_wagon                 -0.8310      0.144     -5.787      0.000      -1.115      -0.547
enginetype_dohcv              -0.7660      0.322     -2.379      0.019      -1.402      -0.130
enginetype_l                   0.3194      0.128      2.492      0.014       0.066       0.573
enginetype_ohc                 0.3475      0.103      3.359      0.001       0.143       0.552
enginetype_ohcf                0.3228      0.122      2.651      0.009       0.082       0.563
cylindernumber_five           -0.3396      0.134     -2.543      0.012      -0.603      -0.076
cylindernumber_four           -0.5050      0.099     -5.083      0.000      -0.701      -0.309
==============================================================================
Omnibus:                       38.750   Durbin-Watson:                   2.106
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               99.305
Skew:                           0.967   Prob(JB):                     2.73e-22
Kurtosis:                       6.285   Cond. No.                         28.1
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [83]:

```python
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe1.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe1.values, i) for i in range(X_train_rfe1.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[83]:

	Features	VIF
10	enginetype_ohc	18.21
13	cylindernumber_four	14.58
0	curbweight	9.12
6	carbody_sedan	6.03
2	carwidth	5.22
1	horsepower	4.50
5	carbody_hatchback	4.07
11	enginetype_ohcf	2.79
7	carbody_wagon	2.77
3	Cars_Category_TopNotch_Cars	2.49
12	cylindernumber_five	2.40
9	enginetype_l	2.14
8	enginetype_dohcv	1.48
4	carbody_hardtop	1.42

In [85]:

```python
# Dropping highly correlated variables and insignificant variables

X_train_rfe2 = X_train_rfe1.drop('enginetype_ohc', 1,)

# Adding a constant variable and Build a sixth fitted model
X_train_rfe2c = sm.add_constant(X_train_rfe2)
lm_rfe2 = sm.OLS(y_train, X_train_rfe2c).fit()
```

```python
#Summary of linear model
print(lm_rfe2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.932
Model:                            OLS   Adj. R-squared:                  0.927
Method:                 Least Squares   F-statistic:                     159.1
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           1.11e-80
Time:                        01:32:09   Log-Likelihood:                -11.812
No. Observations:                 164   AIC:                             51.62
Df Residuals:                     150   BIC:                             95.02
Df Model:                          13
Covariance Type:            nonrobust
=============================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------
const                         0.7112      0.149      4.786      0.000       0.418       1.005
curbweight                    0.1863      0.063      2.965      0.004       0.062       0.310
horsepower                    0.2396      0.047      5.143      0.000       0.148       0.332
carwidth                      0.2138      0.049      4.338      0.000       0.116       0.311
Cars_Category_TopNotch_Cars   1.2196      0.093     13.157      0.000       1.036       1.403
carbody_hardtop              -0.4772      0.183     -2.602      0.010      -0.840      -0.115
carbody_hatchback            -0.6985      0.133     -5.243      0.000      -0.962      -0.435
carbody_sedan                -0.5673      0.131     -4.336      0.000      -0.826      -0.309
carbody_wagon                -0.6923      0.142     -4.869      0.000      -0.973      -0.411
enginetype_dohcv             -0.8977      0.330     -2.717      0.007      -1.550      -0.245
enginetype_l                  0.0663      0.107      0.619      0.537      -0.145       0.278
enginetype_ohcf               0.0164      0.083      0.197      0.844      -0.148       0.181
cylindernumber_five          -0.1006      0.117     -0.861      0.391      -0.331       0.130
cylindernumber_four          -0.3004      0.081     -3.702      0.000      -0.461      -0.140
==============================================================================
Omnibus:                       55.063   Durbin-Watson:                   2.113
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              199.402
Skew:                           1.247   Prob(JB):                     5.02e-44
Kurtosis:                       7.792   Cond. No.                         26.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [86]:

```python
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe2.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe2.values, i) for i in range(X_train_rfe2.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[86]:

	Features	VIF
0	curbweight	8.72
12	cylindernumber_four	8.31
2	carwidth	5.21
6	carbody_sedan	4.71
1	horsepower	4.46
5	carbody_hatchback	3.40
7	carbody_wagon	2.35
3	Cars_Category_TopNotch_Cars	1.94
11	cylindernumber_five	1.71
8	enginetype_dohcv	1.46
9	enginetype_l	1.39
4	carbody_hardtop	1.29

	Features	VIF
10	enginetype_ohcf	1.22

In [90]:

```
# Dropping highly correlated variables and insignificant variables

X_train_rfe3 = X_train_rfe2.drop('curbweight', 1,)

# Adding a constant variable and Build a sixth fitted model
X_train_rfe3c = sm.add_constant(X_train_rfe3)
lm_rfe3 = sm.OLS(y_train, X_train_rfe3c).fit()

#Summary of linear model
print(lm_rfe3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.928
Model:                            OLS   Adj. R-squared:                  0.923
Method:                 Least Squares   F-statistic:                     163.2
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           5.95e-80
Time:                        01:42:01   Log-Likelihood:                -16.481
No. Observations:                 164   AIC:                             58.96
Df Residuals:                     151   BIC:                             99.26
Df Model:                          12
Covariance Type:            nonrobust
==================================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                           0.7402      0.152      4.868      0.000       0.440       1.041
horsepower                      0.3118      0.041      7.654      0.000       0.231       0.392
carwidth                        0.3136      0.037      8.495      0.000       0.241       0.387
Cars_Category_TopNotch_Cars     1.2871      0.092     13.968      0.000       1.105       1.469
carbody_hardtop                -0.5184      0.188     -2.764      0.006      -0.889      -0.148
carbody_hatchback              -0.7483      0.136     -5.521      0.000      -1.016      -0.481
carbody_sedan                  -0.5849      0.134     -4.364      0.000      -0.850      -0.320
carbody_wagon                  -0.6392      0.145     -4.420      0.000      -0.925      -0.353
enginetype_dohcv               -1.2870      0.311     -4.140      0.000      -1.901      -0.673
enginetype_l                    0.1690      0.104      1.626      0.106      -0.036       0.374
enginetype_ohcf                -0.0252      0.084     -0.299      0.766      -0.192       0.141
cylindernumber_five            -0.1433      0.119     -1.205      0.230      -0.378       0.092
cylindernumber_four            -0.3196      0.083     -3.854      0.000      -0.484      -0.156
==============================================================================
Omnibus:                       53.158   Durbin-Watson:                   2.165
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              173.733
Skew:                           1.241   Prob(JB):                     1.88e-38
Kurtosis:                       7.390   Cond. No.                         21.5
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [91]:

```
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe3.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe3.values, i) for i in range(X_train_rfe3.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[91]:

	Features	VIF
11	cylindernumber_four	8.29
5	carbody_sedan	4.71
4	carbody_hatchback	3.34

	Features	VIF
0	horsepower	3.04
1	carwidth	2.78
6	carbody_wagon	2.16
2	Cars_Category_TopNotch_Cars	1.79
10	cylindernumber_five	1.69
3	carbody_hardtop	1.28
8	enginetype_l	1.24
7	enginetype_dohcv	1.22
9	enginetype_ohcf	1.19

In [92]:

```
# Dropping highly correlated variables and insignificant variables

X_train_rfe4 = X_train_rfe3.drop('cylindernumber_four', 1,)

# Adding a constant variable and Build a sixth fitted model
X_train_rfe4c = sm.add_constant(X_train_rfe4)
lm_rfe4 = sm.OLS(y_train, X_train_rfe4c).fit()

#Summary of linear model
print(lm_rfe4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.921
Model:                            OLS   Adj. R-squared:                  0.916
Method:                 Least Squares   F-statistic:                     161.9
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           5.23e-78
Time:                        01:42:52   Log-Likelihood:                -24.174
No. Observations:                 164   AIC:                             72.35
Df Residuals:                     152   BIC:                             109.5
Df Model:                          11
Covariance Type:            nonrobust
===============================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
const                           0.4469      0.137      3.250      0.001       0.175       0.719
horsepower                      0.3905      0.037     10.605      0.000       0.318       0.463
carwidth                        0.3060      0.039      7.945      0.000       0.230       0.382
Cars_Category_TopNotch_Cars     1.3838      0.093     14.940      0.000       1.201       1.567
carbody_hardtop                -0.5369      0.196     -2.742      0.007      -0.924      -0.150
carbody_hatchback              -0.7064      0.141     -5.006      0.000      -0.985      -0.428
carbody_sedan                  -0.5578      0.140     -3.990      0.000      -0.834      -0.282
carbody_wagon                  -0.6143      0.151     -4.070      0.000      -0.912      -0.316
enginetype_dohcv               -1.4839      0.320     -4.633      0.000      -2.117      -0.851
enginetype_l                    0.1676      0.109      1.543      0.125      -0.047       0.382
enginetype_ohcf                -0.0525      0.088     -0.598      0.551      -0.226       0.121
cylindernumber_five             0.0630      0.111      0.568      0.571      -0.156       0.282
==============================================================================
Omnibus:                       47.405   Durbin-Watson:                   2.125
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              144.510
Skew:                           1.121   Prob(JB):                     4.17e-32
Kurtosis:                       7.015   Cond. No.                         18.8
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [93]:

```
# Create a dataframe that will contain the names of all the feature variables and their respective
VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe4.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe4.values, i) for i in range(X_train_rfe4.shape[1
])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[93]:

	Features	VIF
1	carwidth	2.77
0	horsepower	2.56
2	Cars_Category_TopNotch_Cars	1.75
10	cylindernumber_five	1.44
5	carbody_sedan	1.34
8	enginetype_l	1.24
7	enginetype_dohcv	1.20
6	carbody_wagon	1.18
9	enginetype_ohcf	1.17
4	carbody_hatchback	1.14
3	carbody_hardtop	1.07

In [94]:

```
# Dropping highly correlated variables and insignificant variables

X_train_rfe5 = X_train_rfe4.drop('cylindernumber_five', 1,)

# Adding a constant variable and Build a sixth fitted model
X_train_rfe5c = sm.add_constant(X_train_rfe5)
lm_rfe5 = sm.OLS(y_train, X_train_rfe5c).fit()

#Summary of linear model
print(lm_rfe5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.921
Model:                            OLS   Adj. R-squared:                  0.916
Method:                 Least Squares   F-statistic:                     178.9
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           4.47e-79
Time:                        01:52:14   Log-Likelihood:                -24.348
No. Observations:                 164   AIC:                             70.70
Df Residuals:                     153   BIC:                             104.8
Df Model:                          10
Covariance Type:            nonrobust
===============================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
const                           0.4514      0.137      3.296      0.001       0.181       0.722
horsepower                      0.3864      0.036     10.724      0.000       0.315       0.458
carwidth                        0.3159      0.034      9.234      0.000       0.248       0.384
Cars_Category_TopNotch_Cars     1.3839      0.092     14.975      0.000       1.201       1.567
carbody_hardtop                -0.5349      0.195     -2.738      0.007      -0.921      -0.149
carbody_hatchback              -0.7065      0.141     -5.017      0.000      -0.985      -0.428
carbody_sedan                  -0.5577      0.140     -3.998      0.000      -0.833      -0.282
carbody_wagon                  -0.6130      0.151     -4.071      0.000      -0.910      -0.315
enginetype_dohcv               -1.4993      0.318     -4.708      0.000      -2.128      -0.870
enginetype_l                    0.1540      0.106      1.457      0.147      -0.055       0.363
enginetype_ohcf                -0.0538      0.088     -0.615      0.539      -0.227       0.119
==============================================================================
Omnibus:                       45.396   Durbin-Watson:                   2.125
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              133.164
Skew:                           1.085   Prob(JB):                     1.21e-29
Kurtosis:                       6.845   Cond. No.                         18.7
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [95]:

```
# Create a dataframe that will contain the names of all the feature variables and their respective
```

```
VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe5.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe5.values, i) for i in range(X_train_rfe5.shape[1
])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[95]:

	Features	VIF
0	horsepower	2.47
1	carwidth	2.20
2	Cars_Category_TopNotch_Cars	1.75
5	carbody_sedan	1.28
7	enginetype_dohcv	1.20
8	enginetype_l	1.18
9	enginetype_ohcf	1.17
6	carbody_wagon	1.15
4	carbody_hatchback	1.10
3	carbody_hardtop	1.07

In [97]:

```
# Dropping highly correlated variables and insignificant variables

X_train_rfe6 = X_train_rfe5.drop('enginetype_ohcf', 1,)

# Adding a constant variable and Build a sixth fitted model
X_train_rfe6c = sm.add_constant(X_train_rfe6)
lm_rfe6 = sm.OLS(y_train, X_train_rfe6c).fit()

#Summary of linear model
print(lm_rfe6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.921
Model:                            OLS   Adj. R-squared:                  0.916
Method:                 Least Squares   F-statistic:                     199.5
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           3.71e-80
Time:                        01:55:38   Log-Likelihood:                -24.550
No. Observations:                 164   AIC:                             69.10
Df Residuals:                     154   BIC:                             100.1
Df Model:                           9
Covariance Type:            nonrobust
===============================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
const                           0.4438      0.136      3.260      0.001       0.175       0.713
horsepower                      0.3841      0.036     10.740      0.000       0.313       0.455
carwidth                        0.3193      0.034      9.477      0.000       0.253       0.386
Cars_Category_TopNotch_Cars     1.3819      0.092     14.993      0.000       1.200       1.564
carbody_hardtop                -0.5375      0.195     -2.757      0.007      -0.923      -0.152
carbody_hatchback              -0.6999      0.140     -4.995      0.000      -0.977      -0.423
carbody_sedan                  -0.5541      0.139     -3.983      0.000      -0.829      -0.279
carbody_wagon                  -0.6162      0.150     -4.103      0.000      -0.913      -0.320
enginetype_dohcv               -1.4954      0.318     -4.706      0.000      -2.123      -0.868
enginetype_l                    0.1569      0.105      1.489      0.138      -0.051       0.365
==============================================================================
Omnibus:                       45.581   Durbin-Watson:                   2.123
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              133.693
Skew:                           1.090   Prob(JB):                     9.31e-30
Kurtosis:                       6.849   Cond. No.                         18.7
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [98]:

```python
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe6.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe6.values, i) for i in range(X_train_rfe6.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[98]:

	Features	VIF
0	horsepower	2.44
1	carwidth	2.13
2	Cars_Category_TopNotch_Cars	1.74
5	carbody_sedan	1.25
7	enginetype_dohcv	1.19
8	enginetype_l	1.18
4	carbody_hatchback	1.10
6	carbody_wagon	1.07
3	carbody_hardtop	1.06

In [100]:

```python
# Dropping highly correlated variables and insignificant variables

X_train_rfe7 = X_train_rfe6.drop('enginetype_l', 1,)

# Adding a constant variable and Build a sixth fitted model
X_train_rfe7c = sm.add_constant(X_train_rfe7)
lm_rfe7 = sm.OLS(y_train, X_train_rfe7c).fit()

#Summary of linear model
print(lm_rfe7.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.920
Model:                            OLS   Adj. R-squared:                  0.916
Method:                 Least Squares   F-statistic:                     222.4
Date:                Fri, 18 Sep 2020   Prob (F-statistic):           7.19e-81
Time:                        01:58:02   Log-Likelihood:                -25.723
No. Observations:                 164   AIC:                             69.45
Df Residuals:                     155   BIC:                             97.34
Df Model:                           8
Covariance Type:            nonrobust
===============================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
const                           0.4581      0.136      3.360      0.001       0.189       0.727
horsepower                      0.3770      0.036     10.595      0.000       0.307       0.447
carwidth                        0.3330      0.033     10.227      0.000       0.269       0.397
Cars_Category_TopNotch_Cars     1.3642      0.092     14.867      0.000       1.183       1.545
carbody_hardtop                -0.5379      0.196     -2.749      0.007      -0.924      -0.151
carbody_hatchback              -0.7072      0.141     -5.030      0.000      -0.985      -0.429
carbody_sedan                  -0.5583      0.140     -3.999      0.000      -0.834      -0.282
carbody_wagon                  -0.6112      0.151     -4.055      0.000      -0.909      -0.313
enginetype_dohcv               -1.4923      0.319     -4.678      0.000      -2.122      -0.862
==============================================================================
Omnibus:                       45.459   Durbin-Watson:                   2.106
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              124.397
Skew:                           1.116   Prob(JB):                     9.72e-28
Kurtosis:                       6.637   Cond. No.                         18.7
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [101]:

```
# Create a dataframe that will contain the names of all the feature variables and their respective
VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe7.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe7.values, i) for i in range(X_train_rfe7.shape[1
])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[101]:

	Features	VIF
0	horsepower	2.40
1	carwidth	1.98
2	Cars_Category_TopNotch_Cars	1.72
5	carbody_sedan	1.21
7	enginetype_dohcv	1.19
4	carbody_hatchback	1.09
3	carbody_hardtop	1.05
6	carbody_wagon	1.02

In [ ]:

```
##### Now the VIFs and p-values both are within an acceptable range.
##### So we can go ahead and make our predictions using model lm_rfe7 with 8 predictor variable ##
####
```

In [102]:

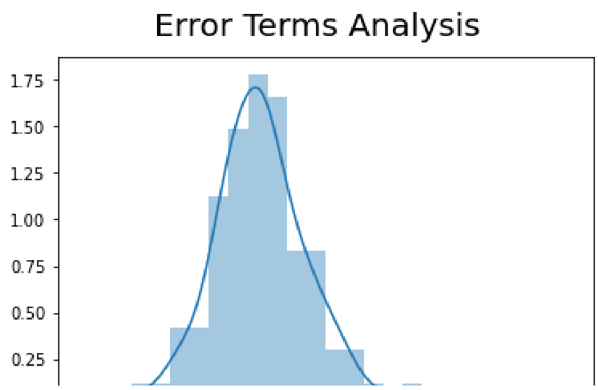
```
######### residual analysis of train data ###########

y_train_price=lm_rfe7.predict(X_train_rfe7c)
```

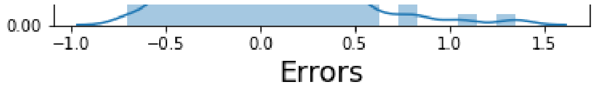
In [103]:

```
###### plot histogram of error terms #########
fig= plt.figure()
sns.distplot((y_train-y_train_price),bins=20)
fig.suptitle('Error Terms Analysis', fontsize = 20)
plt.xlabel('Errors', fontsize = 18)
```

Out[103]:

```
Text(0.5, 0, 'Errors')
```





In [104]:

```
##### making prediction using final model ##########
import warnings
warnings.filterwarnings("ignore")

df_test[sig_num_col] = scaler.transform(df_test[sig_num_col])
df_test.shape
```

Out[104]:

```
(41, 32)
```

In [105]:

```
#### Dividing test set into X_test and y_test

y_test = df_test.pop('price')
X_test = df_test
```

In [106]:

```
# Adding constant
X_test_1 = sm.add_constant(X_test)

X_test_new = X_test_1[X_train_rfe7c.columns]
```

In [107]:

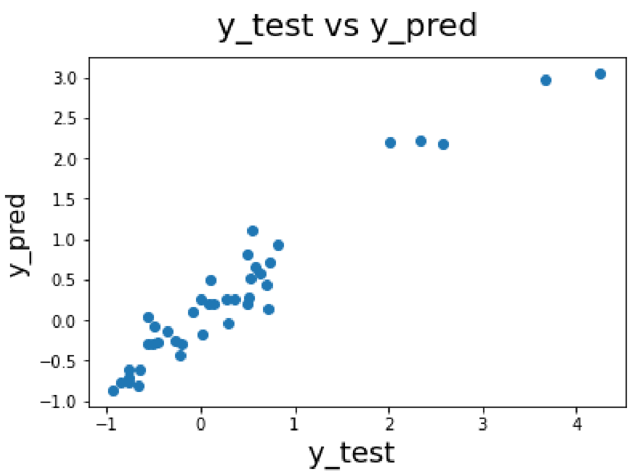
```
# Making predictions using the final model
y_pred = lm_rfe7.predict(X_test_new)
```

In [108]:

```
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)
plt.xlabel('y_test ', fontsize=18)
plt.ylabel('y_pred', fontsize=16)
```

Out[108]:

```
Text(0, 0.5, 'y_pred')
```



In [109]:

```
##### RMSE Square ########

r2_score(y_test, y_pred)
```

Out[109]:

```
0.9164390304661909
```

In [ ]:

```
#### The R2 score of Training set is 0.912 and Test set is 0.909 which is quite close.
#### Hence, We can say that our model is good enough to predict the Car prices using below predict
or variables
####  horsepower
#carwidth
#Cars_Category_TopNotch_Cars
#carbody_sedan
#enginetype_dohcv
#carbody_hatchback
#carbody_hardtop
#carbody_wagon
```

In [ ]:

```
### Model I Conclusions:
### R-sqaured and Adjusted R-squared - 0.92 and 0.91 - 90% variance explained.
### p-values - p-values for all the coefficients seem to be less than the significance level of 0.
05. - meaning that all the
### predictors are statistically significant.
```

In [ ]:

```

```