In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

1.Understanding the data

In [2]:
```
car=pd.read_csv('C:/Users/User/Desktop/datascience/carprice.csv')
car.head()
```

Out[2]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engine
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	

5 rows × 26 columns

In [3]:
```
cars=car.select_dtypes(include=['float64','int64'])
cars.head()
```

Out[3]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio
0	1	3	88.6	168.8	64.1	48.8	2548	130	3.47
1	2	3	88.6	168.8	64.1	48.8	2548	130	3.47
2	3	1	94.5	171.2	65.5	52.4	2823	152	2.68
3	4	2	99.8	176.6	66.2	54.3	2337	109	3.19
4	5	2	99.4	176.6	66.4	54.3	2824	136	3.19

In [4]:
```
# dropping symboling and car_ID as symboling is more of categorical variable as c
#an index type variable and not a predictor
cars= car.drop(['symboling', 'car_ID'], axis=1)
cars.head()
```

Out[4]:

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

5 rows × 24 columns

In [5]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
car_ID              205 non-null int64
symboling           205 non-null int64
CarName             205 non-null object
fueltype            205 non-null object
aspiration          205 non-null object
doornumber          205 non-null object
carbody             205 non-null object
drivewheel          205 non-null object
enginelocation      205 non-null object
wheelbase           205 non-null float64
carlength           205 non-null float64
carwidth            205 non-null float64
carheight           205 non-null float64
curbweight          205 non-null int64
enginetype          205 non-null object
cylindernumber      205 non-null object
enginesize          205 non-null int64
fuelsystem          205 non-null object
boreratio           205 non-null float64
stroke              205 non-null float64
compressionratio    205 non-null float64
horsepower          205 non-null int64
peakrpm             205 non-null int64
citympg             205 non-null int64
highwaympg          205 non-null int64
price               205 non-null float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.7+ KB
```

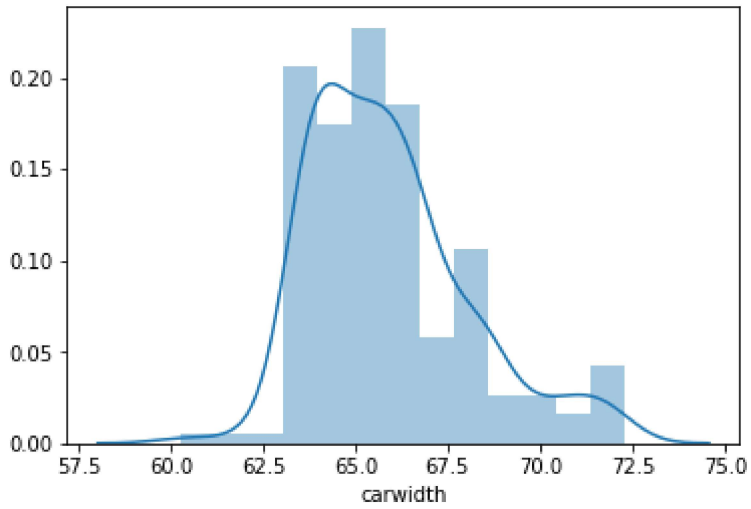
In [6]:
```
car['stroke'].astype('category').value_counts()
```

Out[6]:
```
3.400    20
3.030    14
3.150    14
3.230    14
3.390    13
2.640    11
3.290     9
3.350     9
3.460     8
3.110     6
3.190     6
3.270     6
3.410     6
3.500     6
3.580     6
3.070     6
3.520     5
3.640     5
3.470     4
3.255     4
3.540     4
3.860     4
3.900     3
2.900     3
3.080     2
4.170     2
3.100     2
2.190     2
2.680     2
2.800     2
2.360     1
3.210     1
2.760     1
2.870     1
3.120     1
3.160     1
2.070     1
Name: stroke, dtype: int64
```

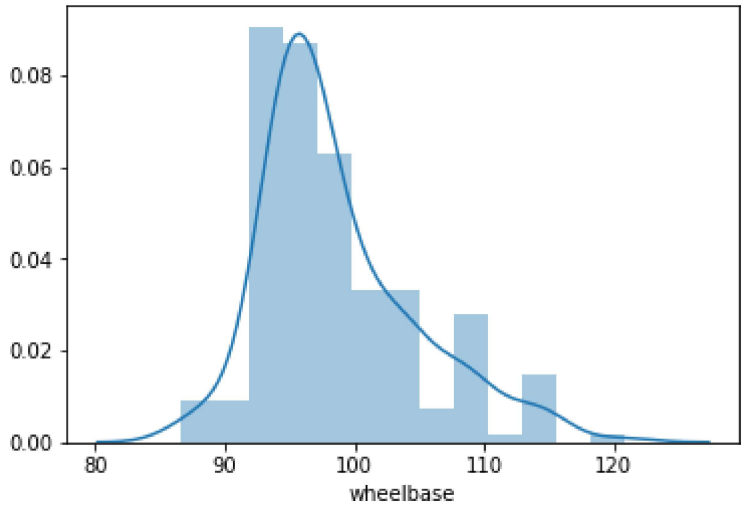
In [7]:
```
car['carbody'].astype('category').value_counts()
```

Out[7]:
```
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: carbody, dtype: int64
```

In [8]:
```
sns.distplot(car['carwidth'])
plt.show()
```

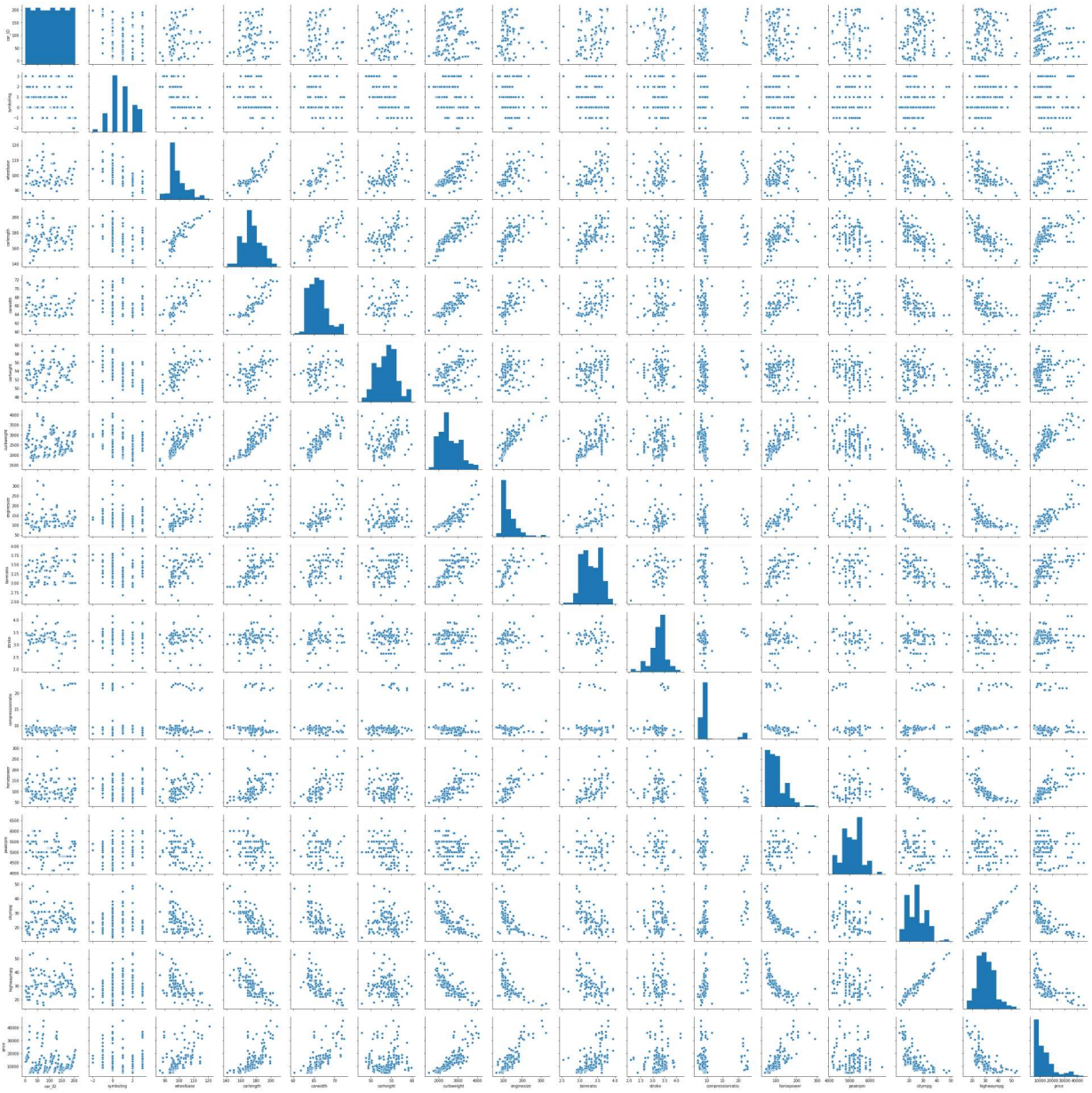


In [9]:
```
sns.distplot(car['wheelbase'])
plt.show()
```

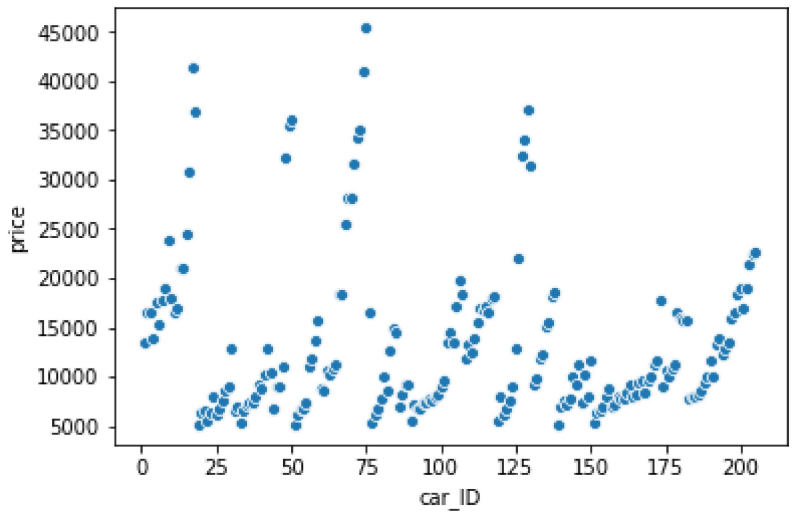


2.Visualising the Data

In [10]:
```
sns.pairplot(car)
plt.show()
```

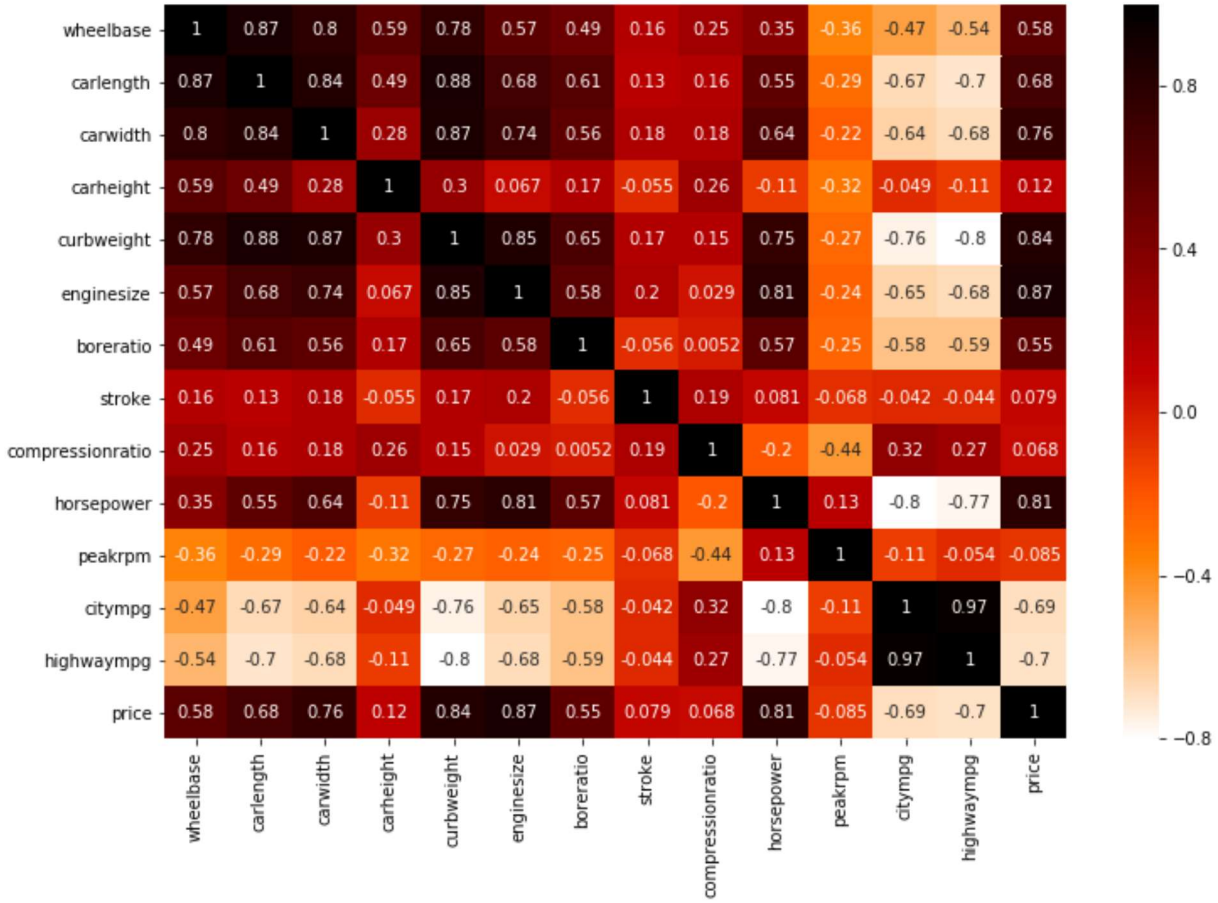


In [11]:
```
for i, col in enumerate (car.columns):
    plt.figure(i)
    sns.scatterplot(x=car[col],y=car['price'])
```



In [12]:
```
corr=cars.corr()  #corealtion with Dependent var and independent var's

plt.figure(figsize=(12,8))
sns.heatmap(corr,annot=True,cmap="gist_heat_r")
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x2270c56da58>



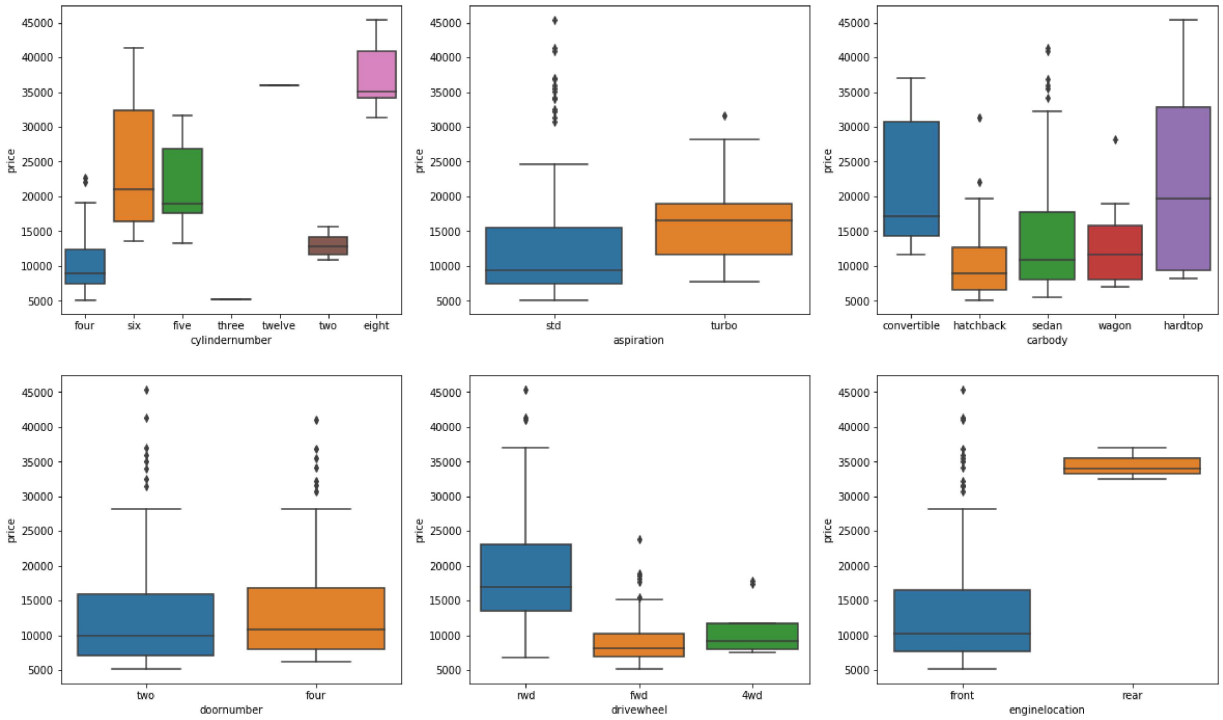
In [13]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
car_ID              205 non-null int64
symboling           205 non-null int64
CarName             205 non-null object
fueltype            205 non-null object
aspiration          205 non-null object
doornumber          205 non-null object
carbody             205 non-null object
drivewheel          205 non-null object
enginelocation      205 non-null object
wheelbase           205 non-null float64
carlength           205 non-null float64
carwidth            205 non-null float64
carheight           205 non-null float64
curbweight          205 non-null int64
enginetype          205 non-null object
cylindernumber      205 non-null object
enginesize          205 non-null int64
fuelsystem          205 non-null object
boreratio           205 non-null float64
stroke              205 non-null float64
compressionratio    205 non-null float64
horsepower          205 non-null int64
peakrpm             205 non-null int64
citympg             205 non-null int64
highwaympg          205 non-null int64
price               205 non-null float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.7+ KB
```

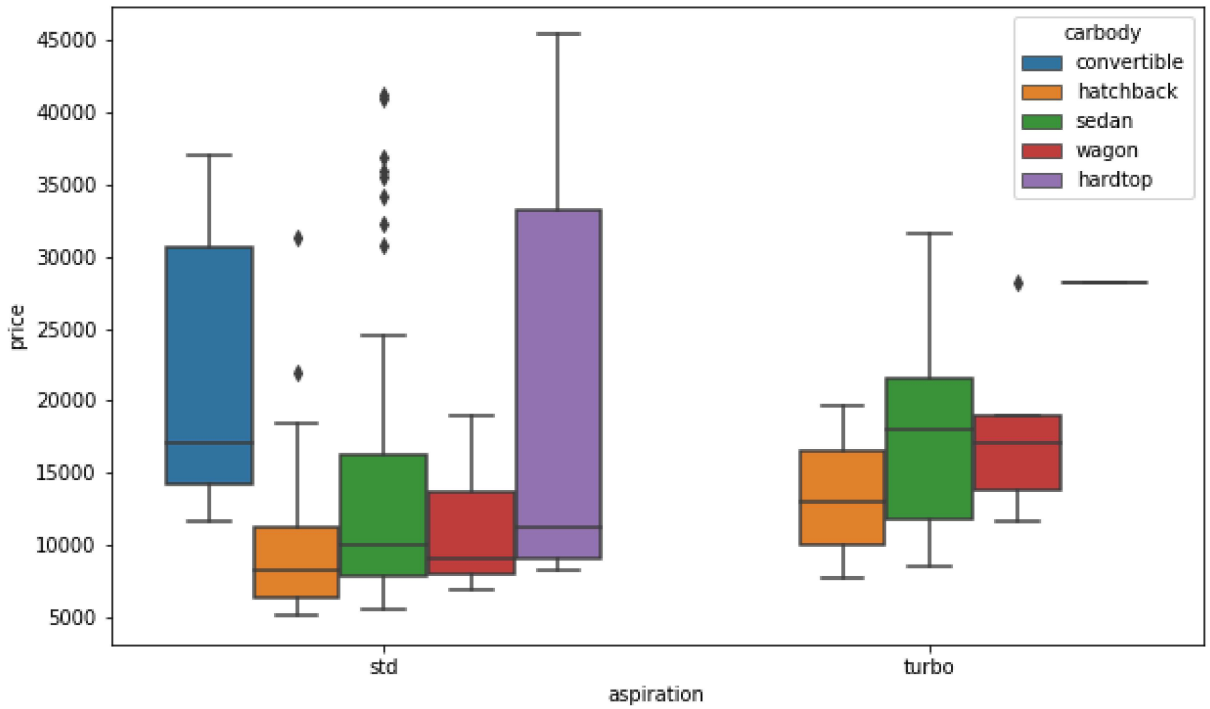
In [14]:
```
## car names only
carnames = car['CarName'].apply(lambda x: x.split(" ")[0])
carnames[:15]
```

Out[14]:
```
0     alfa-romero
1     alfa-romero
2     alfa-romero
3            audi
4            audi
5            audi
6            audi
7            audi
8            audi
9            audi
10            bmw
11            bmw
12            bmw
13            bmw
14            bmw
Name: CarName, dtype: object
```

In [15]:
```
plt.figure(figsize=(20, 12))
plt.subplot(2,3,1)
sns.boxplot(x = 'cylindernumber', y = 'price', data = car)
plt.subplot(2,3,2)
sns.boxplot(x = 'aspiration', y = 'price', data = car)
plt.subplot(2,3,3)
sns.boxplot(x = 'carbody', y = 'price', data = car)
plt.subplot(2,3,4)
sns.boxplot(x = 'doornumber', y = 'price', data = car)
plt.subplot(2,3,5)
sns.boxplot(x = 'drivewheel', y = 'price', data = car)
plt.subplot(2,3,6)
sns.boxplot(x = 'enginelocation', y = 'price', data = car)
plt.show()
```



In [16]:
```
plt.figure(figsize = (10, 6))
sns.boxplot(x = 'aspiration', y = 'price', hue = 'carbody',data = car)
plt.show()
```



In [17]:
```
car['drivewheel'] = car['drivewheel'].astype('object')
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
car_ID              205 non-null int64
symboling           205 non-null int64
CarName             205 non-null object
fueltype            205 non-null object
aspiration          205 non-null object
doornumber          205 non-null object
carbody             205 non-null object
drivewheel          205 non-null object
enginelocation      205 non-null object
wheelbase           205 non-null float64
carlength           205 non-null float64
carwidth            205 non-null float64
carheight           205 non-null float64
curbweight          205 non-null int64
enginetype          205 non-null object
cylindernumber      205 non-null object
enginesize          205 non-null int64
fuelsystem          205 non-null object
boreratio           205 non-null float64
stroke              205 non-null float64
compressionratio    205 non-null float64
horsepower          205 non-null int64
peakrpm             205 non-null int64
citympg             205 non-null int64
highwaympg          205 non-null int64
price               205 non-null float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.7+ KB
```

In [18]:
```
car['car_company']=carnames
car['car_company'].value_counts()
```

Out[18]:
```
toyota         31
nissan         17
mazda          15
honda          13
mitsubishi     13
subaru         12
volvo          11
peugeot        11
dodge           9
volkswagen      9
buick           8
bmw             8
audi            7
plymouth        7
saab            6
porsche         4
isuzu           4
jaguar          3
chevrolet       3
alfa-romero     3
renault         2
vw              2
maxda           2
mercury         1
porcshce        1
toyouta         1
Nissan          1
vokswagen       1
Name: car_company, dtype: int64
```

In [19]:
```
#volkswagen
car.loc[(car['car_company']=="vw")|(car['car_company']=="vokswagen"),"car_company

#porsche

car.loc[(car['car_company']=="porcshce"),"car_company"]="porsche"

#toyota
car.loc[(car['car_company']=="toyouta"),"car_company"]="toyota"

# nissan
car.loc[car['car_company'] == "Nissan", 'car_company'] = 'nissan'

# mazda
car.loc[car['car_company'] == "maxda", 'car_company'] = 'mazda'

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

Out[19]:
```
toyota         32
nissan         18
mazda          17
honda          13
mitsubishi     13
subaru         12
volkswagen     12
peugeot        11
volvo          11
dodge           9
buick           8
bmw             8
plymouth        7
audi            7
saab            6
porsche         5
isuzu           4
chevrolet       3
alfa-romero     3
jaguar          3
renault         2
mercury         1
Name: car_company, dtype: int64
```

3.Data Preparation

In [20]:
```
x=car.drop(columns=['price',"car_ID"])
y=car['price']
```

In [21]:
```
y=car['price']
y.head()
```

Out[21]:
```
0    13495.0
1    16500.0
2    16500.0
3    13950.0
4    17450.0
Name: price, dtype: float64
```

In [22]:
```
cars_categorical = x.select_dtypes(include=['object'])
cars_categorical.head()
```

Out[22]:

	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype
0	alfa-romero giulia	gas	std	two	convertible	rwd	front	dohc
1	alfa-romero stelvio	gas	std	two	convertible	rwd	front	dohc
2	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	ohcv
3	audi 100 ls	gas	std	four	sedan	fwd	front	ohc
4	audi 100ls	gas	std	four	sedan	4wd	front	ohc

In [23]:
```
cars_dummies = pd.get_dummies(cars_categorical, drop_first=False)
cars_dummies.head()
```

Out[23]:

	CarName_Nissan versa	CarName_alfa-romero Quadrifoglio	CarName_alfa-romero giulia	CarName_alfa-romero stelvio	CarName_audi 100 ls	CarName_audi 100ls
0	0	0	1	0	0	0
1	0	0	0	1	0	0
2	0	1	0	0	0	0
3	0	0	0	0	1	0
4	0	0	0	0	0	1

5 rows × 207 columns

In [24]:
```
cars_dummies = pd.get_dummies(cars_categorical, drop_first=True)
cars_dummies.head()
```

Out[24]:

	CarName_alfa-romero Quadrifoglio	CarName_alfa-romero giulia	CarName_alfa-romero stelvio	CarName_audi 100 ls	CarName_audi 100ls	CarName_audi 4000	Ca
0	0	1	0	0	0	0	
1	0	0	1	0	0	0	
2	1	0	0	0	0	0	
3	0	0	0	1	0	0	
4	0	0	0	0	1	0	

5 rows × 196 columns

In [25]:
```
x=x.drop(columns=cars_categorical)
x.head()
```

Out[25]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke
0	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68
1	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68
2	1	94.5	171.2	65.5	52.4	2823	152	2.68	3.47
3	2	99.8	176.6	66.2	54.3	2337	109	3.19	3.40
4	2	99.4	176.6	66.4	54.3	2824	136	3.19	3.40

In [26]:
```
x=pd.concat([x,cars_dummies],axis=1)
x.head()
```

Out[26]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke
0	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68
1	3	88.6	168.8	64.1	48.8	2548	130	3.47	2.68
2	1	94.5	171.2	65.5	52.4	2823	152	2.68	3.47
3	2	99.8	176.6	66.2	54.3	2337	109	3.19	3.40
4	2	99.4	176.6	66.4	54.3	2824	136	3.19	3.40

5 rows × 210 columns

In [27]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Columns: 210 entries, symboling to car_company_volvo
dtypes: float64(7), int64(7), uint8(196)
memory usage: 61.7 KB
```

In [28]:
```
x.columns
```

Out[28]:
```
Index(['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight',
       'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio',
       ...
       'car_company_nissan', 'car_company_peugeot', 'car_company_plymouth',
       'car_company_porsche', 'car_company_renault', 'car_company_saab',
       'car_company_subaru', 'car_company_toyota', 'car_company_volkswagen',
       'car_company_volvo'],
      dtype='object', length=210)
```

4: Splitting the Data into Training and Testing Sets

In [29]:
```
from sklearn.model_selection import train_test_split

# We specify this so that the train and test data set always have the same rows,
np.random.seed(0)
x_train, y_test = train_test_split(cars, train_size = 0.7, test_size = 0.3, rand
```

In [30]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [31]:
```
scaler = MinMaxScaler()
```

In [32]:
```
num_vars = ['curbweight','carlength','enginesize','horsepower','price']

x_train[num_vars] = scaler.fit_transform(x_train[num_vars])
```

In [33]:
```
x_train.head()
```

Out[33]:

	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase
122	plymouth fury gran sedan	gas	std	four	sedan	fwd	front	93.7
125	porsche macan	gas	std	two	hatchback	rwd	front	94.5
166	toyota corolla tercel	gas	std	two	hatchback	rwd	front	94.5
1	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6
199	volvo diesel	gas	turbo	four	wagon	rwd	front	104.3

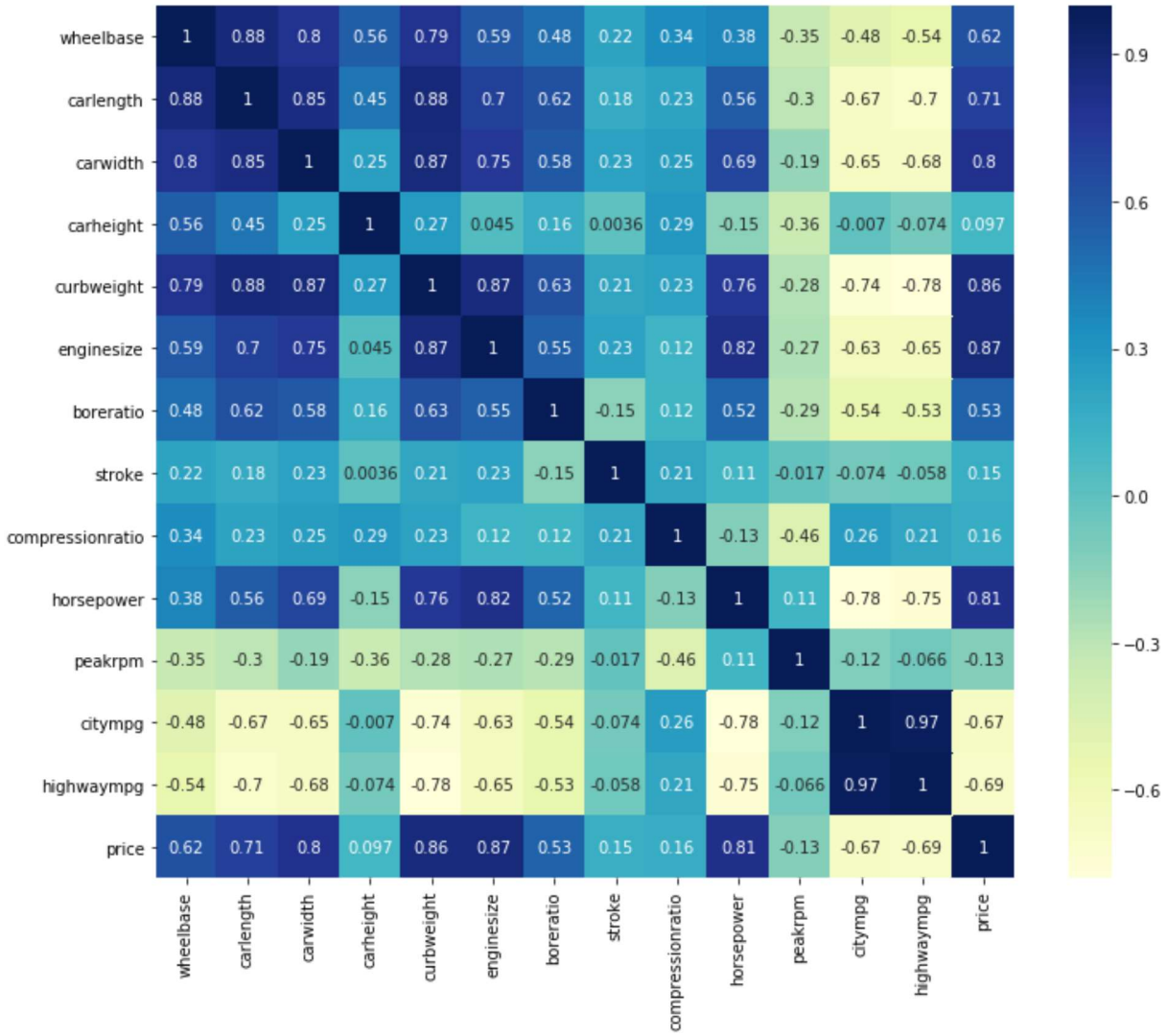
5 rows × 24 columns

In [34]:
```
x_train.describe()
```

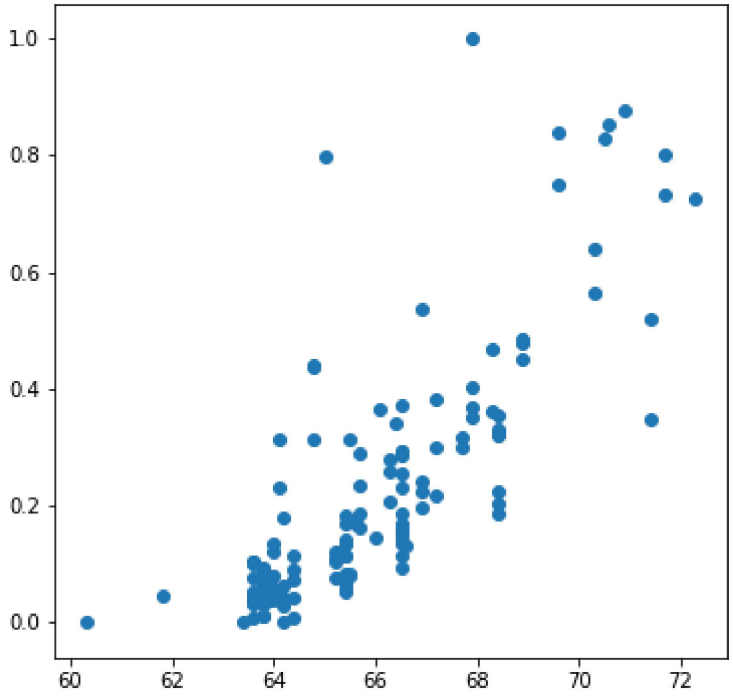
Out[34]:

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	st
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.00
mean	98.523077	0.525476	65.839860	53.551748	0.407878	0.241351	3.307413	3.25
std	5.961835	0.204848	2.214203	2.433766	0.211269	0.154619	0.260997	0.31
min	86.600000	0.000000	60.300000	47.800000	0.000000	0.000000	2.680000	2.19
25%	94.500000	0.399187	63.950000	51.800000	0.245539	0.135849	3.065000	3.11
50%	96.500000	0.502439	65.400000	53.700000	0.355702	0.184906	3.310000	3.27
75%	101.200000	0.669919	66.900000	55.350000	0.559542	0.301887	3.540000	3.40
max	115.600000	1.000000	72.300000	59.100000	1.000000	1.000000	3.940000	4.17

In [35]:
```
plt.figure(figsize = (12, 10))
sns.heatmap(x_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



In [36]:
```
plt.figure(figsize=[6,6])
plt.scatter(x_train.carwidth, x_train.price)
plt.show()
```



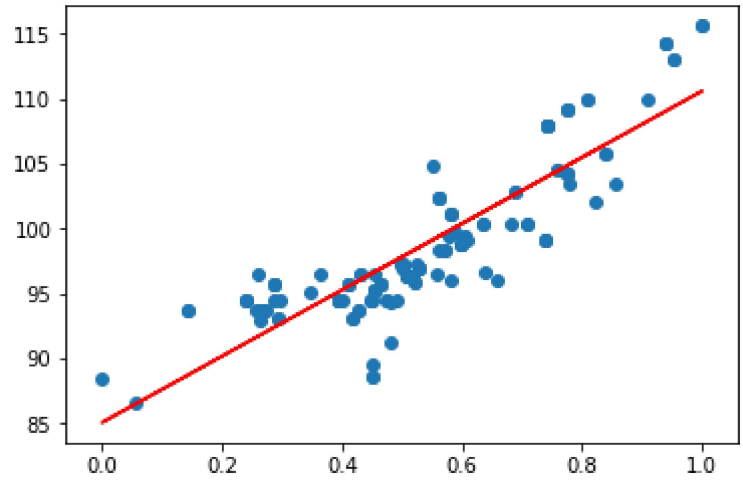
In [44]:
```
y_train = x_train.pop('wheelbase')
x_train = x_train
```

In [45]:
```
import statsmodels.api as sm

# Add a constant
x_train_lm = sm.add_constant(x_train[['carlength']])

# Create a first fitted model
lr = sm.OLS(y_train, x_train_lm).fit()
```

In [46]:
```
lr.params
```

Out[46]:
```
const        85.096919
carlength    25.550462
dtype: float64
```

In [47]:
```
plt.scatter(x_train_lm.iloc[:, 1], y_train)
plt.plot(x_train_lm.iloc[:, 1], 85.09+ 25.5*x_train_lm.iloc[:, 1], 'r')
plt.show()
```



In [48]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              wheelbase   R-squared:                       0.771
Model:                            OLS   Adj. R-squared:                  0.769
Method:                 Least Squares   F-statistic:                     474.0
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           6.13e-47
Time:                        15:11:06   Log-Likelihood:                -352.41
No. Observations:                 143   AIC:                             708.8
Df Residuals:                     141   BIC:                             714.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         85.0969      0.662    128.624      0.000      83.789      86.405
carlength     25.5505      1.174     21.771      0.000      23.230      27.871
==============================================================================
Omnibus:                        0.533   Durbin-Watson:                   1.967
Prob(Omnibus):                  0.766   Jarque-Bera (JB):                0.518
Skew:                          -0.143   Prob(JB):                        0.772
Kurtosis:                       2.925   Cond. No.                         6.30
==============================================================================

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

In [49]:
```
x_train_lm = x_train[['carlength', 'carwidth']]
```

In [50]:
```
import statsmodels.api as sm
x_train_lm = sm.add_constant(x_train_lm)

lr = sm.OLS(y_train, x_train_lm).fit()

lr.params
```

Out[50]:
```
const        53.877754
carlength    20.871260
carwidth      0.511513
dtype: float64
```

In [51]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              wheelbase   R-squared:                       0.781
Model:                            OLS   Adj. R-squared:                  0.778
Method:                 Least Squares   F-statistic:                     249.6
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           6.85e-47
Time:                        15:11:08   Log-Likelihood:                -349.14
No. Observations:                 143   AIC:                             704.3
Df Residuals:                     140   BIC:                             713.2
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         53.8778     12.220      4.409      0.000      29.719      78.036
carlength     20.8713      2.161      9.658      0.000      16.599      25.144
carwidth       0.5115      0.200      2.558      0.012       0.116       0.907
==============================================================================
Omnibus:                        2.042   Durbin-Watson:                   1.945
Prob(Omnibus):                  0.360   Jarque-Bera (JB):                1.692
Skew:                          -0.259   Prob(JB):                        0.429
Kurtosis:                       3.123   Cond. No.                     3.46e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The condition number is large, 3.46e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [52]:
```
x_train_lm = x_train[['carlength', 'carwidth','carheight']]
```

In [53]:
```
import statsmodels.api as sm
x_train_lm = sm.add_constant(x_train_lm)

lr = sm.OLS(y_train, x_train_lm).fit()

lr.params
```

Out[53]:
```
const         2.070431
carlength    14.465968
carwidth      0.846456
carheight     0.618478
dtype: float64
```

In [54]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              wheelbase   R-squared:                       0.827
Model:                            OLS   Adj. R-squared:                  0.823
Method:                 Least Squares   F-statistic:                     221.7
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           9.02e-53
Time:                        15:11:11   Log-Likelihood:                -332.21
No. Observations:                 143   AIC:                             672.4
Df Residuals:                     139   BIC:                             684.3
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.0704     13.819      0.150      0.881     -25.253      29.394
carlength     14.4660      2.195      6.591      0.000      10.126      18.805
carwidth       0.8465      0.187      4.538      0.000       0.478       1.215
carheight      0.6185      0.102      6.093      0.000       0.418       0.819
==============================================================================
Omnibus:                        0.904   Durbin-Watson:                   1.921
Prob(Omnibus):                  0.636   Jarque-Bera (JB):                0.993
Skew:                          -0.180   Prob(JB):                        0.609
Kurtosis:                       2.806   Cond. No.                     5.65e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The condition number is large, 5.65e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [55]:
```
cars.columns
```

Out[55]:
```
Index(['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 [56]:
```
car.car_company.describe()
```

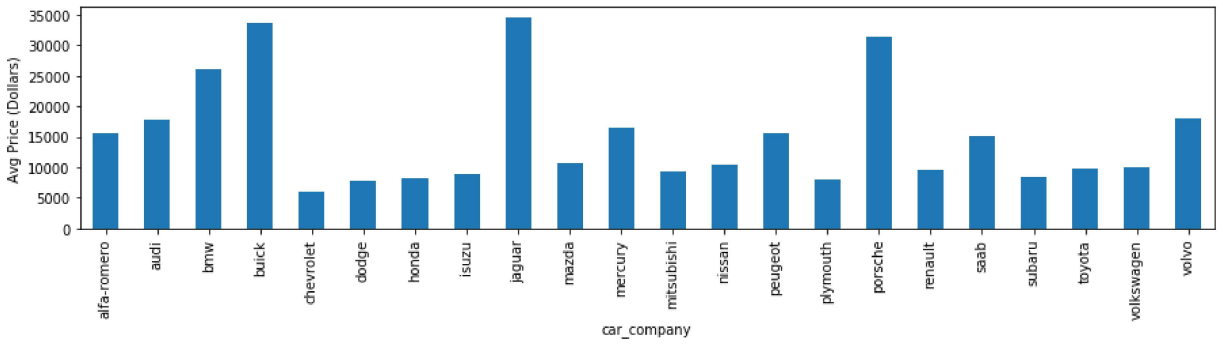
Out[56]:
```
count        205
unique        22
top       toyota
freq          32
Name: car_company, dtype: object
```

In [57]:
```
car.price.describe()
```

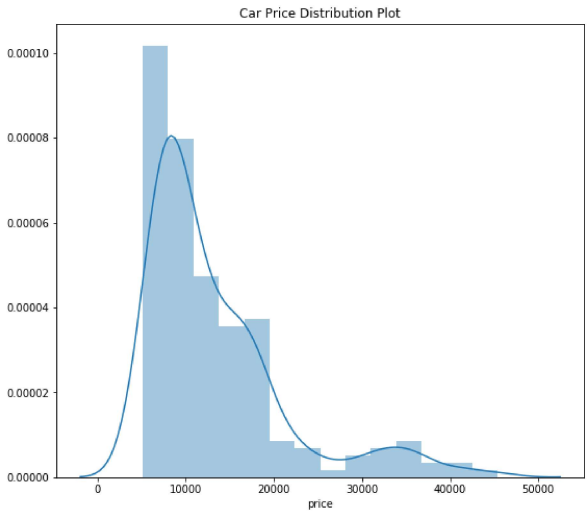
Out[57]:
```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
25%       7788.000000
50%      10295.000000
75%      16503.000000
max      45400.000000
Name: price, dtype: float64
```

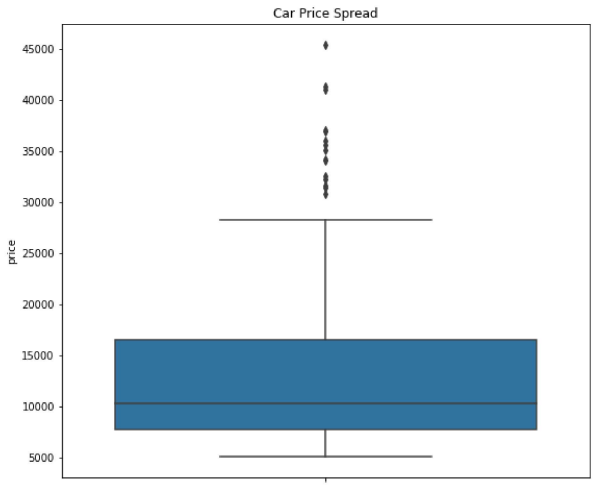
In [58]:
```
car_cmp_avg_price = car[['car_company','price']].groupby("car_company", as_index
plt = car_cmp_avg_price.plot(x = 'car_company', kind='bar',legend = False, sort_
plt.set_xlabel("car_company")
plt.set_ylabel("Avg Price (Dollars)")
```

Out[58]:
```
Text(0, 0.5, 'Avg Price (Dollars)')
```



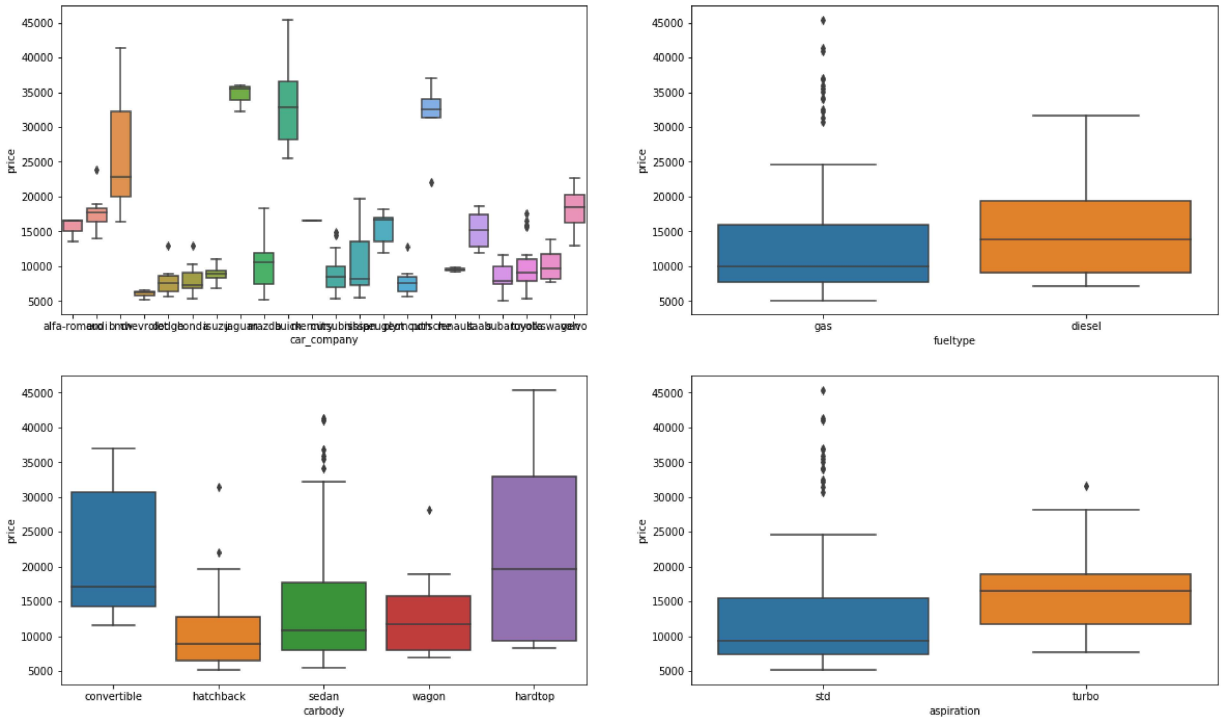
In [59]:
```
from matplotlib import *
import sys
import pylab as pl
pl.figure(figsize=(20,8))

pl.subplot(1,2,1)
pl.title('Car Price Distribution Plot')
sns.distplot(car.price)

pl.subplot(1,2,2)
pl.title('Car Price Spread')
sns.boxplot(y=car.price)

pl.show()
```



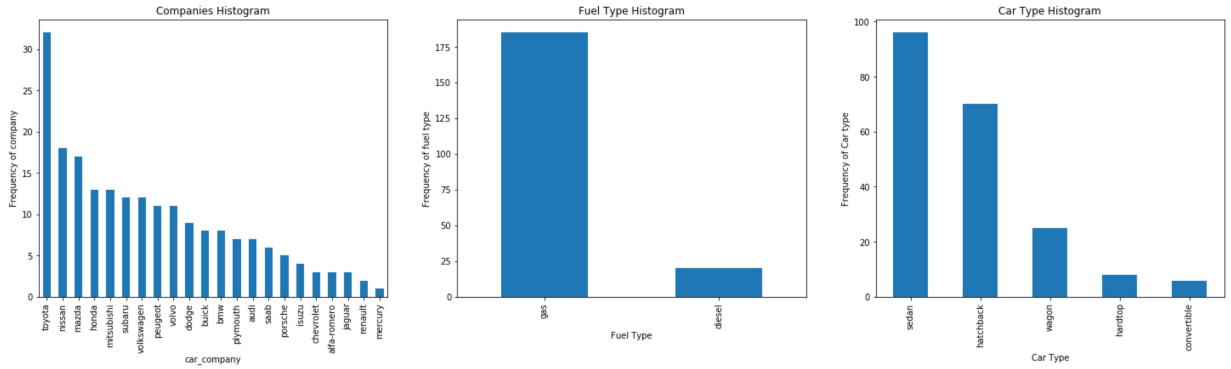


In [60]:
```
pl.figure(figsize=(20, 12))
pl.subplot(2,2,1)
sns.boxplot(x = 'car_company', y = 'price', data = car)
pl.subplot(2,2,2)
sns.boxplot(x = 'fueltype', y = 'price', data = car)
pl.subplot(2,2,3)
sns.boxplot(x = 'carbody', y = 'price', data = car)
pl.subplot(2,2,4)
sns.boxplot(x = 'aspiration', y = 'price', data = car)
```

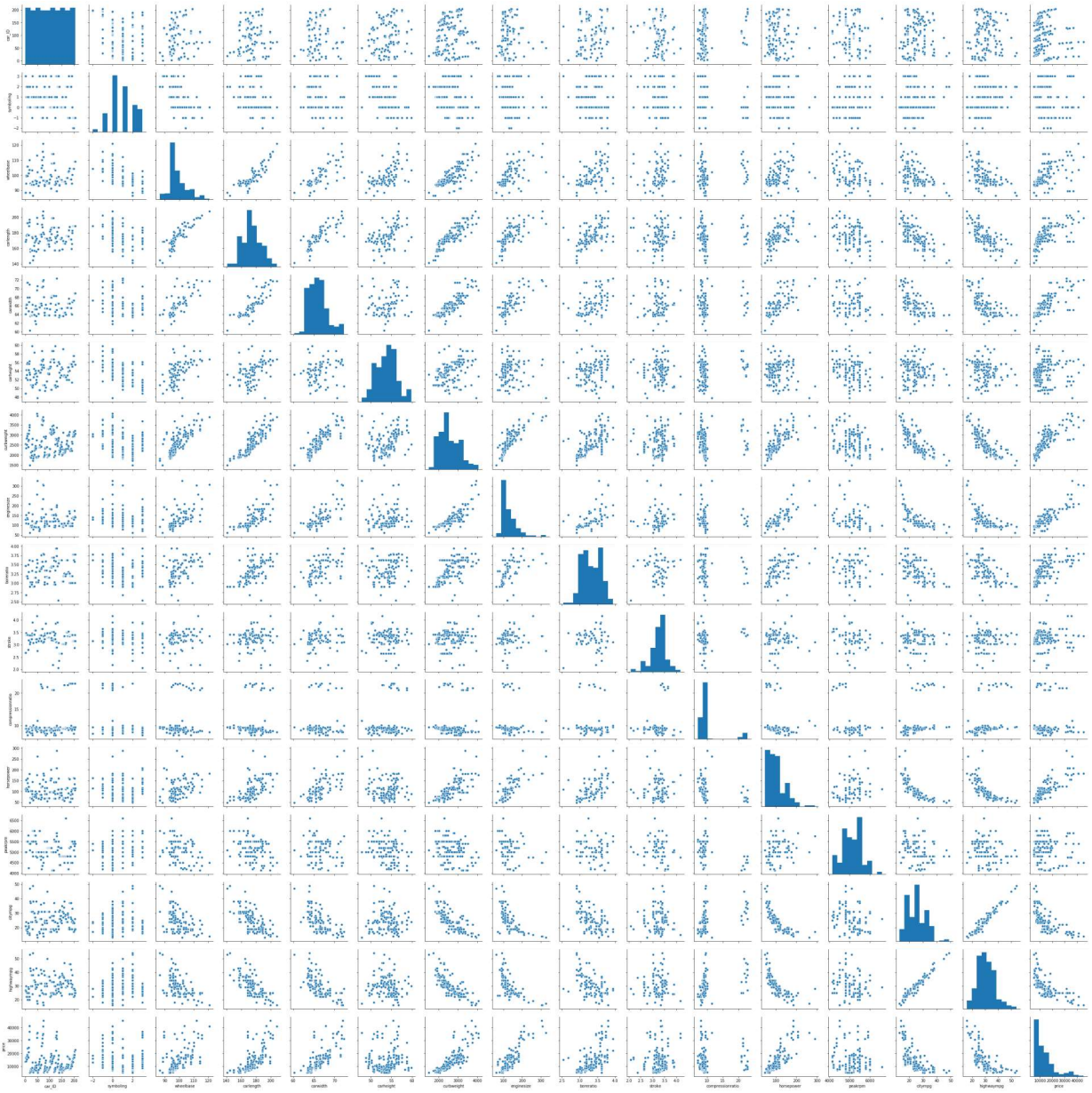
Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x2270b01a7b8>



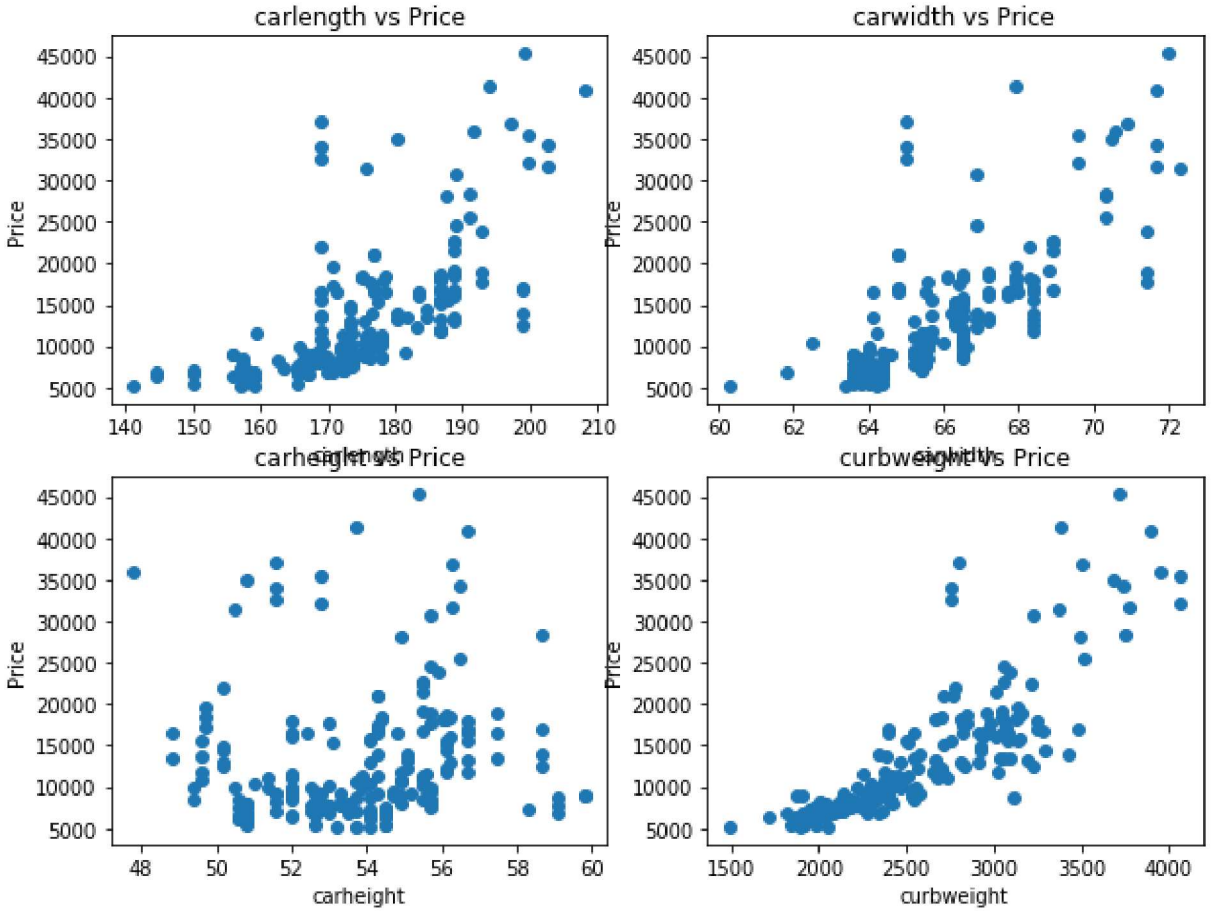
In [61]:
```
pl.figure(figsize=(25, 6))
pl.subplot(1,3,1)
pl1 = car.car_company.value_counts().plot(kind='bar')
pl.title('Companies Histogram')
pl1.set(xlabel = 'car_company', ylabel='Frequency of company')
pl.subplot(1,3,2)
pl1 = car.fueltype.value_counts().plot(kind='bar')
pl.title('Fuel Type Histogram')
pl1.set(xlabel = 'Fuel Type', ylabel='Frequency of fuel type')
pl.subplot(1,3,3)
pl1 = car.carbody.value_counts().plot(kind='bar')
pl.title('Car Type Histogram')
pl1.set(xlabel = 'Car Type', ylabel='Frequency of Car type')
pl.show()
```



In [62]:
```
sns.pairplot(car)
pl.show()
```



In [63]:
```
def scatter(x,fig):
    pl.subplot(5,2,fig)
    pl.scatter(car[x],car['price'])
    pl.title(x+' vs Price')
    pl.ylabel('Price')
    pl.xlabel(x)
pl.figure(figsize=(10,20))
scatter('carlength', 1)
scatter('carwidth', 2)
scatter('carheight', 3)
scatter('curbweight', 4)
```



In [64]:
```
car_lr = car[['price', 'fueltype', 'aspiration','carbody', 'drivewheel','wheelba
                'cylindernumber', 'enginesize', 'boreratio','horsepower',  'carl
car_lr.head()
```

Out[64]:

	price	fueltype	aspiration	carbody	drivewheel	wheelbase	curbweight	enginetype	cyli
0	13495.0	gas	std	convertible	rwd	88.6	2548	dohc	
1	16500.0	gas	std	convertible	rwd	88.6	2548	dohc	
2	16500.0	gas	std	hatchback	rwd	94.5	2823	ohcv	
3	13950.0	gas	std	sedan	fwd	99.8	2337	ohc	
4	17450.0	gas	std	sedan	4wd	99.4	2824	ohc	

Dummy variables

In [65]:
```
status = pd.get_dummies(car['cylindernumber'])
status.head()
```

Out[65]:

	eight	five	four	six	three	twelve	two
0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0
2	0	0	0	1	0	0	0
3	0	0	1	0	0	0	0
4	0	1	0	0	0	0	0

In [66]:
```
status = pd.get_dummies(car['cylindernumber'], drop_first = True)
car= pd.concat([car, status], axis = 1)
car.head()
```

Out[66]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engine
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	

5 rows × 33 columns

In [67]:
```
y_train = x_train.pop('price')
X_train = x_train
```

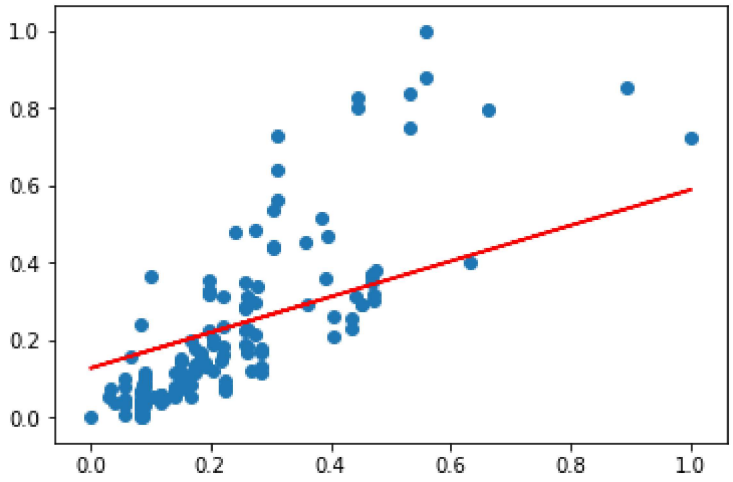
Model building

In [68]:
```
import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train[['horsepower']])
lr = sm.OLS(y_train, X_train_lm).fit()
```

In [69]:
```
lr.params
```

Out[69]:
```
const        -0.019484
horsepower    1.050556
dtype: float64
```

In [70]:
```
pl.scatter(X_train_lm.iloc[:, 1], y_train)
pl.plot(X_train_lm.iloc[:, 1], 0.127 + 0.462*X_train_lm.iloc[:, 1], 'r')
pl.show()
```



In [71]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.650
Model:                            OLS   Adj. R-squared:                  0.647
Method:                 Least Squares   F-statistic:                     261.8
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           6.04e-34
Time:                        15:12:02   Log-Likelihood:                91.997
No. Observations:                 143   AIC:                            -180.0
Df Residuals:                     141   BIC:                            -174.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0195      0.018     -1.068      0.287      -0.056       0.017
horsepower     1.0506      0.065     16.180      0.000       0.922       1.179
==============================================================================
Omnibus:                       33.630   Durbin-Watson:                   1.832
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.578
Skew:                           1.166   Prob(JB):                     2.32e-12
Kurtosis:                       4.886   Cond. No.                         6.38
==============================================================================

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

In [72]:
```
X_train_lm = X_train[['horsepower', 'boreratio']]
```

In [73]:
```
import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train_lm)
lr = sm.OLS(y_train, X_train_lm).fit()
lr.params
```

Out[73]:
```
const        -0.427726
horsepower    0.943668
boreratio     0.130778
dtype: float64
```

In [74]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.668
Model:                            OLS   Adj. R-squared:                  0.664
Method:                 Least Squares   F-statistic:                     141.0
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           2.86e-34
Time:                        15:12:03   Log-Likelihood:                 95.839
No. Observations:                 143   AIC:                            -185.7
Df Residuals:                     140   BIC:                            -176.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.4277      0.148     -2.892      0.004      -0.720      -0.135
horsepower     0.9437      0.074     12.722      0.000       0.797       1.090
boreratio      0.1308      0.047      2.780      0.006       0.038       0.224
==============================================================================
Omnibus:                       33.546   Durbin-Watson:                   1.773
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.888
Skew:                           1.156   Prob(JB):                     1.99e-12
Kurtosis:                       4.923   Cond. No.                         52.8
==============================================================================

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

In [75]:
```
X_train_lm = X_train[['horsepower', 'boreratio','stroke']]
```

In [76]:
```
import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train_lm)
lr = sm.OLS(y_train, X_train_lm).fit()
lr.params
```

Out[76]:
```
const        -0.744309
horsepower    0.908106
boreratio     0.156044
stroke        0.074184
dtype: float64
```

In [77]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.679
Model:                            OLS   Adj. R-squared:                  0.672
Method:                 Least Squares   F-statistic:                     98.00
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           3.95e-34
Time:                        15:12:03   Log-Likelihood:                98.192
No. Observations:                 143   AIC:                            -188.4
Df Residuals:                     139   BIC:                            -176.5
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.7443      0.207     -3.595      0.000      -1.154      -0.335
horsepower     0.9081      0.075     12.098      0.000       0.760       1.057
boreratio      0.1560      0.048      3.258      0.001       0.061       0.251
stroke         0.0742      0.034      2.157      0.033       0.006       0.142
==============================================================================
Omnibus:                       30.003   Durbin-Watson:                   1.740
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.670
Skew:                           1.053   Prob(JB):                     7.34e-11
Kurtosis:                       4.844   Cond. No.                         99.2
==============================================================================

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

Checking VIF

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:

$VIFi=1/1-Ri2$

In [78]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [79]:
```
def build_model(X,y):
    X = sm.add_constant(X) #Adding the constant
    lm = sm.OLS(y,X).fit() # fitting the model
    print(lm.summary()) # model summary
    return X

def checkVIF(X):
    vif = pd.DataFrame()
    vif['Features'] = X.columns
    vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[
    vif['VIF'] = round(vif['VIF'], 2)
    vif = vif.sort_values(by = "VIF", ascending = False)
    return(vif)
```

In [80]:
```
X_train_new = build_model(X_train_lm,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.679
Model:                            OLS   Adj. R-squared:                  0.672
Method:                 Least Squares   F-statistic:                     98.00
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           3.95e-34
Time:                        15:12:04   Log-Likelihood:                 98.192
No. Observations:                 143   AIC:                            -188.4
Df Residuals:                     139   BIC:                            -176.5
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.7443      0.207     -3.595      0.000      -1.154      -0.335
horsepower     0.9081      0.075     12.098      0.000       0.760       1.057
boreratio      0.1560      0.048      3.258      0.001       0.061       0.251
stroke         0.0742      0.034      2.157      0.033       0.006       0.142
==============================================================================
Omnibus:                       30.003   Durbin-Watson:                   1.740
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.670
Skew:                           1.053   Prob(JB):                     7.34e-11
Kurtosis:                       4.844   Cond. No.                         99.2
==============================================================================

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

In [81]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.679
Model:                            OLS   Adj. R-squared:                  0.672
Method:                 Least Squares   F-statistic:                     98.00
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           3.95e-34
Time:                        15:12:04   Log-Likelihood:                98.192
No. Observations:                 143   AIC:                            -188.4
Df Residuals:                     139   BIC:                            -176.5
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.7443      0.207     -3.595      0.000      -1.154      -0.335
horsepower     0.9081      0.075     12.098      0.000       0.760       1.057
boreratio      0.1560      0.048      3.258      0.001       0.061       0.251
stroke         0.0742      0.034      2.157      0.033       0.006       0.142
==============================================================================
Omnibus:                       30.003   Durbin-Watson:                   1.740
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.670
Skew:                           1.053   Prob(JB):                     7.34e-11
Kurtosis:                       4.844   Cond. No.                         99.2
==============================================================================

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

In [82]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.679
Model:                            OLS   Adj. R-squared:                  0.672
Method:                 Least Squares   F-statistic:                     98.00
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           3.95e-34
Time:                        15:12:04   Log-Likelihood:                 98.192
No. Observations:                 143   AIC:                            -188.4
Df Residuals:                     139   BIC:                            -176.5
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.7443      0.207     -3.595      0.000      -1.154      -0.335
horsepower     0.9081      0.075     12.098      0.000       0.760       1.057
boreratio      0.1560      0.048      3.258      0.001       0.061       0.251
stroke         0.0742      0.034      2.157      0.033       0.006       0.142
==============================================================================
Omnibus:                       30.003   Durbin-Watson:                   1.740
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.670
Skew:                           1.053   Prob(JB):                     7.34e-11
Kurtosis:                       4.844   Cond. No.                         99.2
==============================================================================

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

In [83]:
```
checkVIF(X_train_new)
```

Out[83]:

	Features	VIF
0	const	401.88
2	boreratio	1.45
1	horsepower	1.44
3	stroke	1.08

In [84]:
```
X_train_new = X_train_new.drop(["stroke"], axis = 1)
```

In [85]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.668
Model:                            OLS   Adj. R-squared:                  0.664
Method:                 Least Squares   F-statistic:                     141.0
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           2.86e-34
Time:                        15:12:04   Log-Likelihood:                 95.839
No. Observations:                 143   AIC:                            -185.7
Df Residuals:                     140   BIC:                            -176.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.4277      0.148     -2.892      0.004      -0.720      -0.135
horsepower     0.9437      0.074     12.722      0.000       0.797       1.090
boreratio      0.1308      0.047      2.780      0.006       0.038       0.224
==============================================================================
Omnibus:                       33.546   Durbin-Watson:                   1.773
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.888
Skew:                           1.156   Prob(JB):                     1.99e-12
Kurtosis:                       4.923   Cond. No.                         52.8
==============================================================================

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

In [86]:
```
checkVIF(X_train_new)
```

Out[86]:

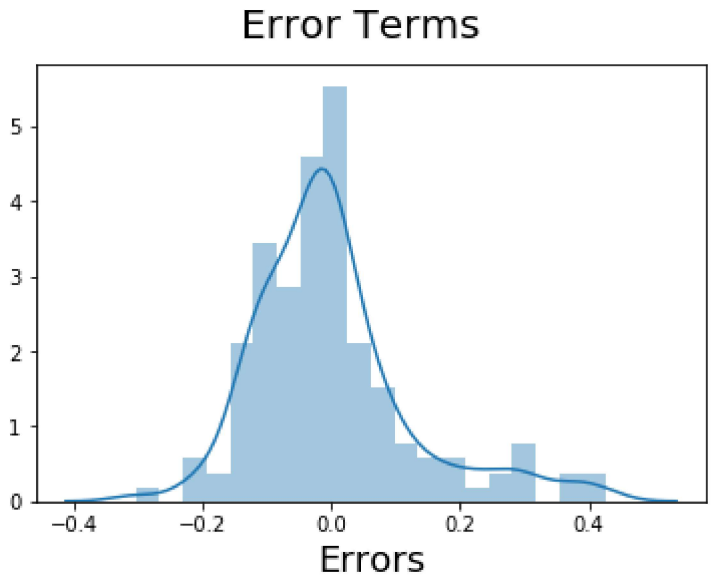
	Features	VIF
0	const	199.87
1	horsepower	1.37
2	boreratio	1.37

Residual analysis of a model

In [87]:
```
lm = sm.OLS(y_train,X_train_new).fit()
y_train_price = lm.predict(X_train_new)
```

In [88]:
```
fig = pl.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)
pl.xlabel('Errors', fontsize = 18)
```

Out[88]:
```
Text(0.5, 0, 'Errors')
```



Model evaluation

In [159]:
```
car=pd.read_csv('C:/Users/User/Desktop/datascience/carprice.csv')
car.head()
```

Out[159]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engine
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	

5 rows × 26 columns

In [160]:
```
from sklearn.model_selection import train_test_split

# We specify this so that the train and test data set always have the same rows,
np.random.seed(0)
x_train, y_test = train_test_split(cars, train_size = 0.7, test_size = 0.3, rand
```

In [161]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [162]:
```
scaler = MinMaxScaler()
```

In [163]:
```
num_vars = ['curbweight','carlength','enginesize','horsepower','price','carwidth

x_train[num_vars] = scaler.fit_transform(x_train[num_vars])
```

In [164]:
```
y_test = x_train.pop('price')
X_test = x_train
```

In [165]:
```
X_train_new = X_train_lm.drop('stroke',axis=1)
X_test_new = X_test[X_train_new.columns]
X_test_new = sm.add_constant(X_test_new)
```

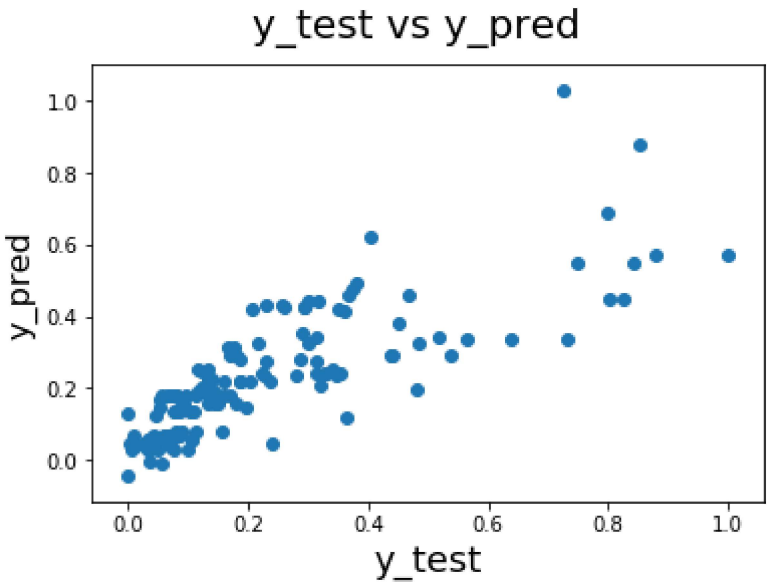
In [166]:
```
y_pred = lm.predict(X_test_new)
```

In [167]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[167]: 0.6682475689627897

In [168]:
```
import matplotlib.pyplot as plt
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[168]:
```
Text(0, 0.5, 'y_pred')
```



In [169]:
```
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.668
Model:                            OLS   Adj. R-squared:                  0.664
Method:                 Least Squares   F-statistic:                     141.0
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           2.86e-34
Time:                        15:45:35   Log-Likelihood:                 95.839
No. Observations:                 143   AIC:                            -185.7
Df Residuals:                     140   BIC:                            -176.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.4277      0.148     -2.892      0.004      -0.720      -0.135
horsepower     0.9437      0.074     12.722      0.000       0.797       1.090
boreratio      0.1308      0.047      2.780      0.006       0.038       0.224
==============================================================================
Omnibus:                       33.546   Durbin-Watson:                   1.773
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.888
Skew:                           1.156   Prob(JB):                     1.99e-12
Kurtosis:                       4.923   Cond. No.                         52.8
==============================================================================

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

In [ ]:

In [ ]: