In [1]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

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

In [3]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [4]:

```
car=pd.read_csv('D:/task/carprice.csv')
car.head()
```

Out[4]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	en
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 [5]:

```
car.shape
```

Out[5]:

```
(205, 26)
```

In [6]:

```
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 [7]:

```
car.describe()
```

Out[7]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	e
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	2
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	1
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	1
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	1
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	3

In [8]:

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

Out[8]:

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

In [9]:

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

Out[9]:

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

5 rows × 24 columns

In [10]:

```
car['symboling'].astype('category').value_counts()
```

Out[10]:

```
 0    67
 1    54
 2    32
 3    27
-1    22
-2     3
Name: symboling, dtype: int64
```

In [11]:

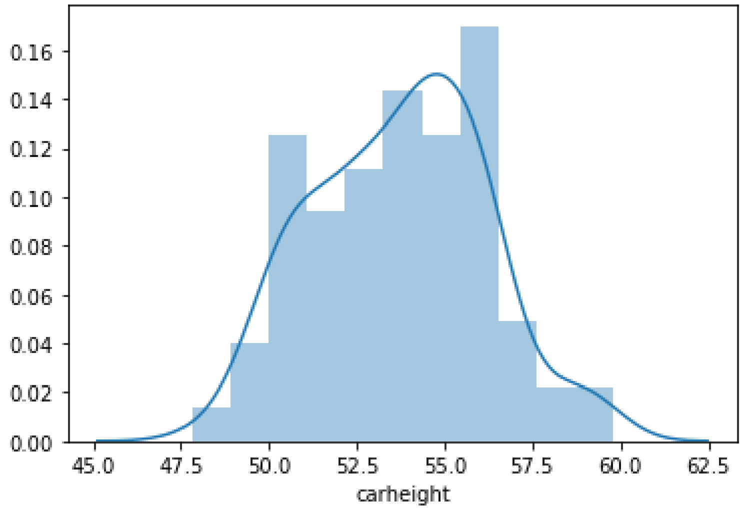
```
car['aspiration'].astype('category').value_counts()
```

Out[11]:

```
std      168
turbo     37
Name: aspiration, dtype: int64
```

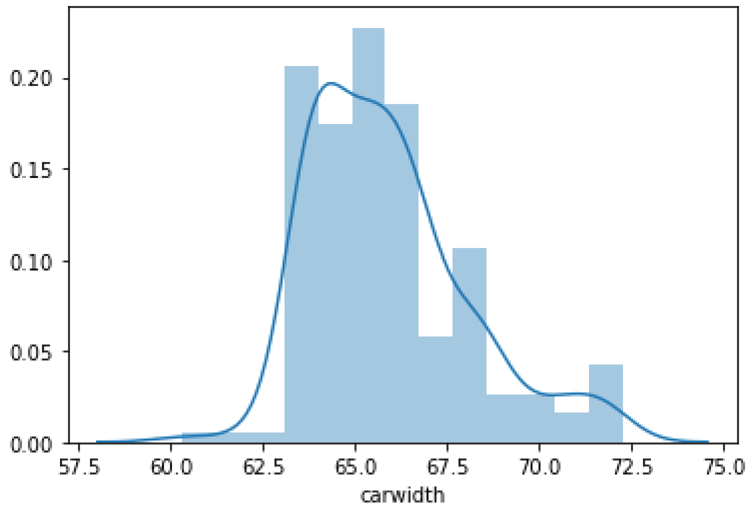
In [12]:

```
sns.distplot(car['carheight'])
plt.show()
```



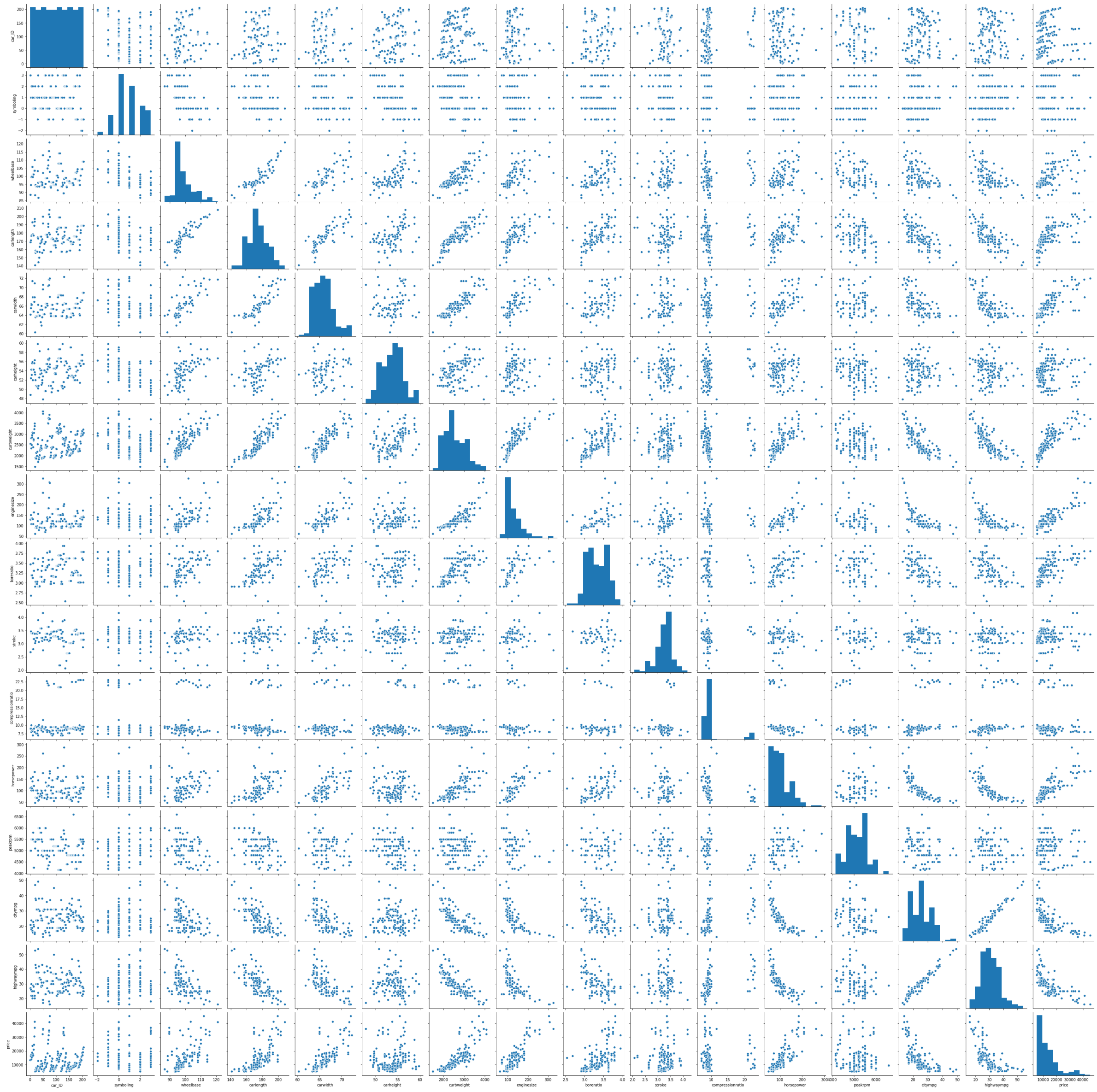
In [13]:

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



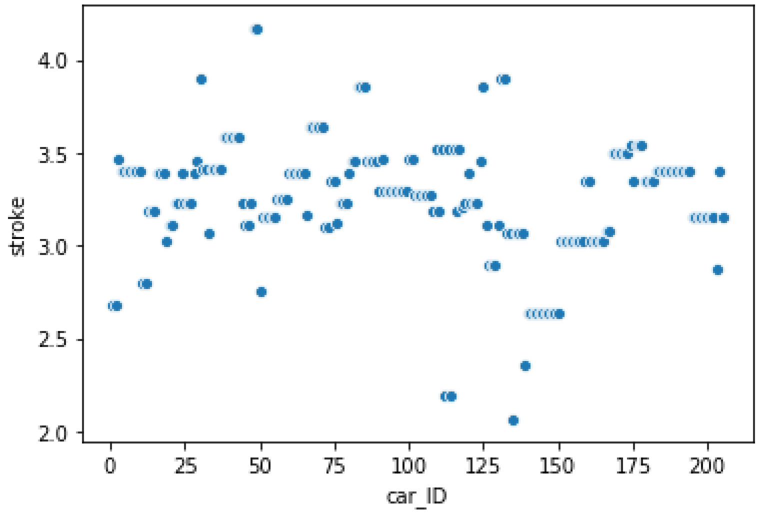
In [14]:

```
#VISUALISING THE DATA
sns.pairplot(car)
plt.show()
```



In [15]:

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

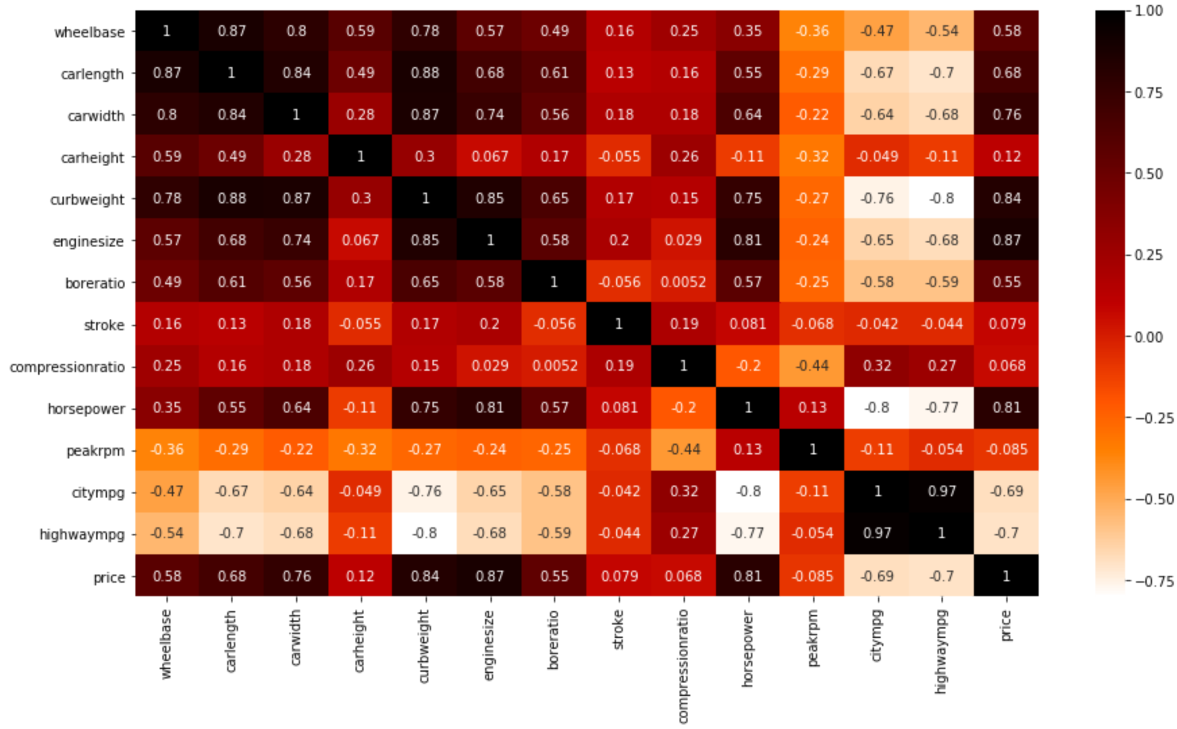


In [16]:

```
#corealtion with Dependent var and independent var's
corr=cars.corr()
plt.figure(figsize=(15,8))
sns.heatmap(corr,annot=True,cmap="gist_heat_r")
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ede939e1c0>
```



In [17]:

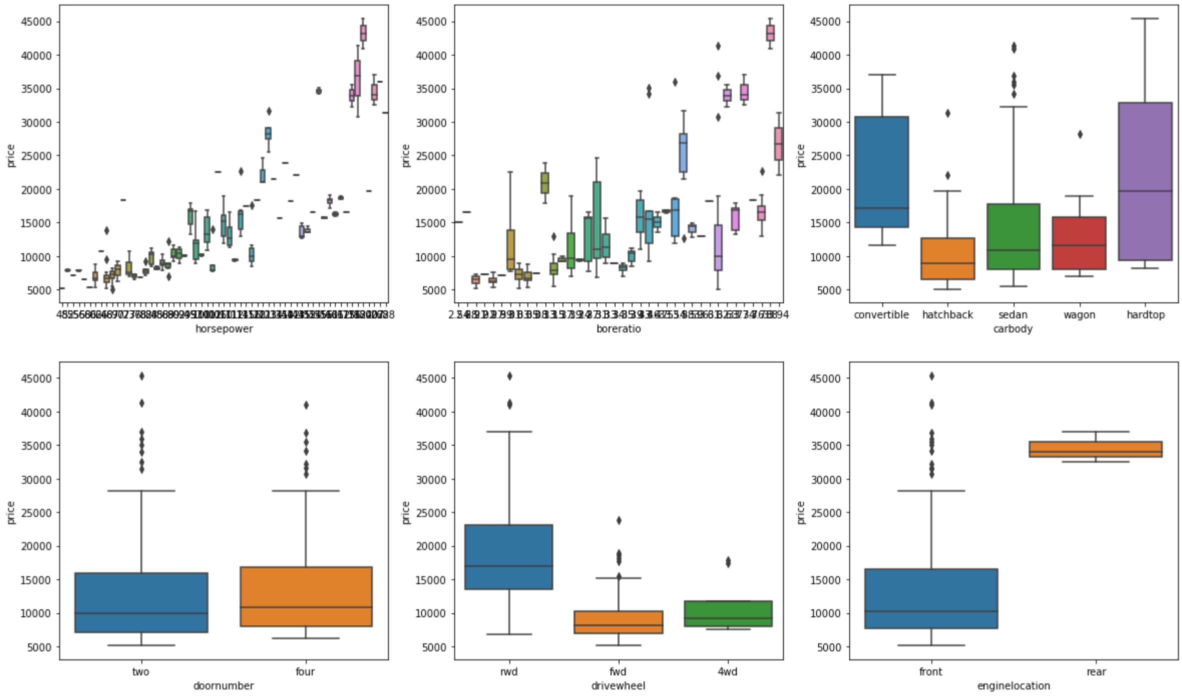
```
##ONLY CAR NAMES
carnames = car['CarName'].apply(lambda x: x.split(" ")[0])
carnames[:21]
```

Out[17]:

```
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
15            bmw
16            bmw
17            bmw
18      chevrolet
19      chevrolet
20      chevrolet
Name: CarName, dtype: object
```

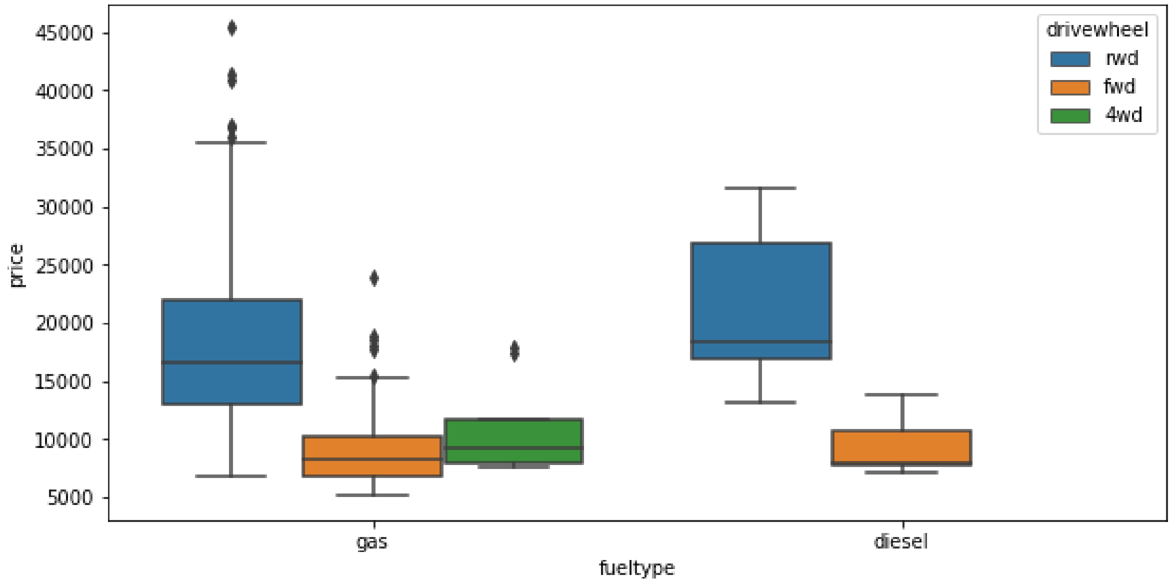
In [18]:

```
plt.figure(figsize=(20, 12))
plt.subplot(2,3,1)
sns.boxplot(x = 'horsepower', y = 'price', data = car)
plt.subplot(2,3,2)
sns.boxplot(x = 'boreratio', 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 [19]:

```
plt.figure(figsize = (10, 5))
sns.boxplot(x = 'fueltype', y = 'price', hue = 'drivewheel',data = car)
plt.show()
```



In [20]:

```
car['symboling'] = car['symboling'].astype('object')
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    object 
 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(7), object(11)
memory usage: 41.8+ KB
```

In [21]:

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

Out[21]:

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

In [22]:

```
#bmw

car.loc[(car['car_company']=="bmw"),"car_company"]="BMW"

#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'] == "audi", 'car_company'] = 'Audi'

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

Out[22]:

```
toyota         32
nissan         18
mazda          15
mitsubishi     13
honda          13
subaru         12
volvo          11
peugeot        11
volkswagen      9
dodge           9
BMW             8
buick           8
plymouth        7
Audi            7
saab            6
isuzu           4
porsche         4
jaguar          3
alfa-romero     3
chevrolet       3
vw              2
renault         2
maxda           2
porcshce        1
vokswagen       1
mercury         1
Name: car_company, dtype: int64
```

In [23]:

```
#DATA PREPARATION
x=car.drop(columns=['price',"car_ID"])
y=car['price']
y.head()
```

Out[23]:

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

In [24]:

```
cars_category = x.select_dtypes(include=['object'])
cars_category.head()
```

Out[24]:

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

In [25]:

```
cars_dummy = pd.get_dummies(cars_category, drop_first=True)
cars_dummy.head()
```

Out[25]:

	symboling_-1	symboling_0	symboling_1	symboling_2	symboling_3	CarName_alfa-romero Quadrifoglio	CarNan romer
0	0	0	0	0	1	0	
1	0	0	0	0	1	0	
2	0	0	1	0	0	1	
3	0	0	0	1	0	0	
4	0	0	0	1	0	0	

5 rows × 205 columns

In [26]:

```
x=x.drop(columns=cars_category)
x.head()
```

Out[26]:

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

In [27]:

```
x=pd.concat([x,cars_dummy],axis=1)
x.head()
```

Out[27]:

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

5 rows × 218 columns

In [28]:

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Columns: 218 entries, wheelbase to car_company_vw
dtypes: float64(7), int64(6), uint8(205)
memory usage: 62.0 KB
```

In [29]:

```
x.columns
```

Out[29]:

```
Index(['wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight',
       'enginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepowe
r',
       ...
       'car_company_porcshce', 'car_company_porsche', 'car_company_renault',
       'car_company_saab', 'car_company_subaru', 'car_company_toyota',
       'car_company_vokswagen', 'car_company_volkswagen', 'car_company_volv
o',
       'car_company_vw'],
      dtype='object', length=218)
```

In [30]:

```
#TRAIN-TEST
from sklearn.model_selection import train_test_split

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

In [31]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [32]:

```
num_vars = ['curbweight','carlength','curbweight','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	wheelb
122	plymouth fury gran sedan	gas	std	four	sedan	fwd	front	
125	porsche macan	gas	std	two	hatchback	rwd	front	
166	toyota corolla tercel	gas	std	two	hatchback	rwd	front	
1	alfa-romero stelvio	gas	std	two	convertible	rwd	front	
199	volvo diesel	gas	turbo	four	wagon	rwd	front	1

5 rows × 24 columns

In [34]:

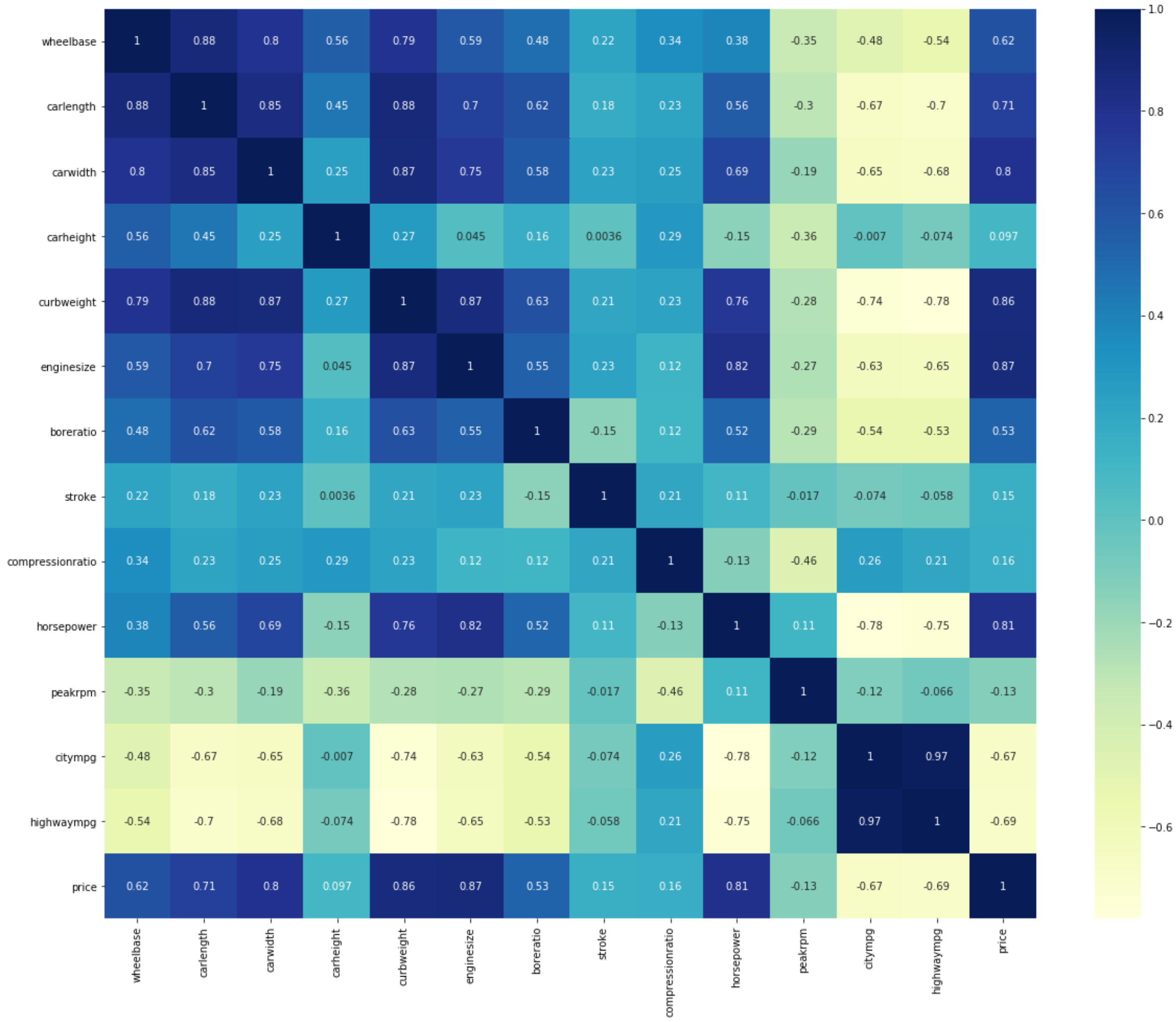
```
x_train.describe()
```

Out[34]:

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	14
mean	98.523077	0.525476	65.839860	53.551748	0.407878	0.241351	3.307413	
std	5.961835	0.204848	2.214203	2.433766	0.211269	0.154619	0.260997	
min	86.600000	0.000000	60.300000	47.800000	0.000000	0.000000	2.680000	
25%	94.500000	0.399187	63.950000	51.800000	0.245539	0.135849	3.065000	
50%	96.500000	0.502439	65.400000	53.700000	0.355702	0.184906	3.310000	
75%	101.200000	0.669919	66.900000	55.350000	0.559542	0.301887	3.540000	
max	115.600000	1.000000	72.300000	59.100000	1.000000	1.000000	3.940000	

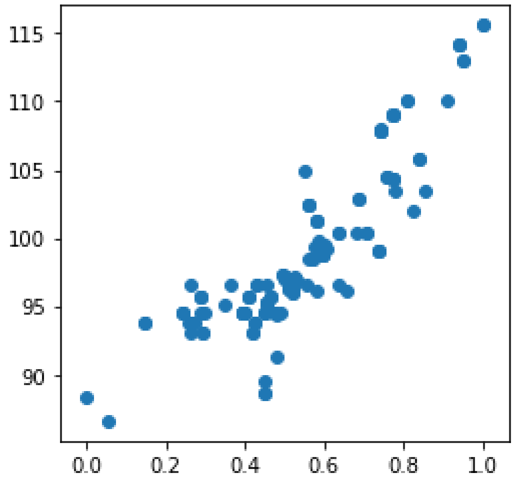
In [35]:

```
plt.figure(figsize = (20, 16))
sns.heatmap(x_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



In [36]:

```
plt.figure(figsize=[4,4])
plt.scatter(x_train.carlength, x_train.wheelbase)
plt.show()
```



In [37]:

```
y_train = x_train.pop('stroke')
x_train = x_train
```

In [38]:

```
import statsmodels.api as sm
```

In [39]:

```
#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 [40]:

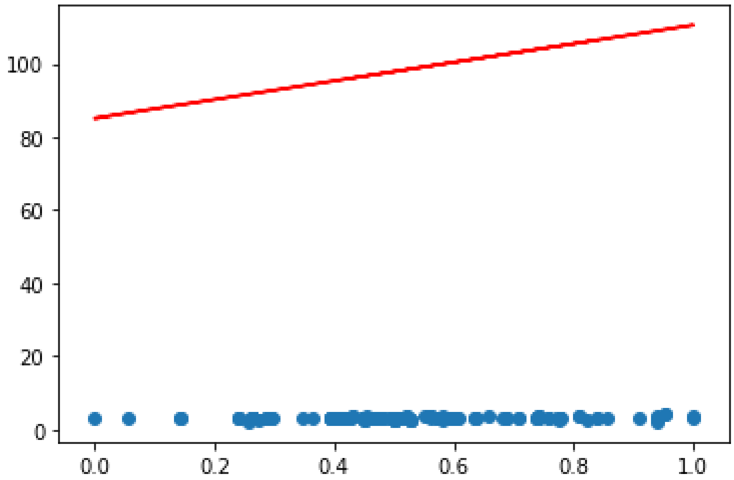
```
lr.params
```

Out[40]:

```
const        3.108318
carlength    0.269760
dtype: float64
```

In [41]:

```
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 [42]:

```
print(lr.summary())
```

```
                            OLS Regression Results
============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.0
31
Model:                            OLS   Adj. R-squared:                  0.0
24
Method:                 Least Squares   F-statistic:                     4.5
50
Date:                Sun, 26 Apr 2020   Prob (F-statistic):             0.03
46
Time:                        09:11:23   Log-Likelihood:                -33.8
18
No. Observations:                 143   AIC:                             71.
64
Df Residuals:                     141   BIC:                             77.
56
Df Model:                           1
Covariance Type:            nonrobust
============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------
--
const          3.1083      0.071     43.601      0.000       2.967       3.2
49
carlength      0.2698      0.126      2.133      0.035       0.020       0.5
20
============================================================================
==
Omnibus:                       16.116   Durbin-Watson:                   2.0
81
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               22.6
20
Skew:                          -0.625   Prob(JB):                     1.22e-
05
Kurtosis:                       4.494   Cond. No.                         6.
30
============================================================================
==

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

In [43]:

```
x_train_lm = x_train[['carlength', 'carwidth']]
```

In [44]:

```
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[44]:

```
const        0.558465
carlength   -0.112418
carwidth     0.041778
dtype: float64
```

In [45]:

```
print(lr.summary())
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.0
56
Model:                            OLS   Adj. R-squared:                  0.0
43
Method:                 Least Squares   F-statistic:                     4.1
62
Date:                Sun, 26 Apr 2020   Prob (F-statistic):             0.01
75
Time:                        09:11:30   Log-Likelihood:                -31.9
59
No. Observations:                 143   AIC:                             69.
92
Df Residuals:                     140   BIC:                             78.
81
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const          0.5585      1.330      0.420      0.675      -2.070       3.1
87
carlength     -0.1124      0.235     -0.478      0.633      -0.577       0.3
53
carwidth       0.0418      0.022      1.920      0.057      -0.001       0.0
85
==============================================================================
==
Omnibus:                       15.116   Durbin-Watson:                   2.0
21
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               20.5
46
Skew:                          -0.604   Prob(JB):                     3.46e-
05
Kurtosis:                       4.411   Cond. No.                     3.46e+
03
==============================================================================
==

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

In [46]:

```
x_train_lm = x_train[['carlength', 'carwidth','carheight']]
```

In [47]:

```
print(lr.summary())
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.0
56
Model:                            OLS   Adj. R-squared:                  0.0
43
Method:                 Least Squares   F-statistic:                     4.1
62
Date:                Sun, 26 Apr 2020   Prob (F-statistic):             0.01
75
Time:                        09:11:32   Log-Likelihood:                -31.9
59
No. Observations:                 143   AIC:                             69.
92
Df Residuals:                     140   BIC:                             78.
81
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const          0.5585      1.330      0.420      0.675      -2.070       3.1
87
carlength     -0.1124      0.235     -0.478      0.633      -0.577       0.3
53
carwidth       0.0418      0.022      1.920      0.057      -0.001       0.0
85
==============================================================================
==
Omnibus:                       15.116   Durbin-Watson:                   2.0
21
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               20.5
46
Skew:                          -0.604   Prob(JB):                     3.46e-
05
Kurtosis:                       4.411   Cond. No.                     3.46e+
03
==============================================================================
==

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

In [48]:

```
cars.columns
```

Out[48]:

```
Index(['CarName', 'fueltype', 'aspiration', 'doornumber', 'carbody',
       'drivewheel', 'enginelocation', 'wheelbase', 'carlength', 'carwidth',
       'carheight', 'curbweight', 'enginetype', 'cylindernumber', 'enginesiz
e',
       'fuelsystem', 'boreratio', 'stroke', 'compressionratio', 'horsepowe
r',
       'peakrpm', 'citympg', 'highwaympg', 'price'],
      dtype='object')
```

In [49]:

```
x_train_lm = x_train[['horsepower', 'boreratio','wheelbase']]
```

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          3.002135
horsepower     0.357183
boreratio     -0.506372
wheelbase      0.018691
dtype: float64
```

In [51]:

```
print(lr.summary())
```

```
                            OLS Regression Results
============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.1
65
Model:                            OLS   Adj. R-squared:                  0.1
47
Method:                 Least Squares   F-statistic:                     9.1
55
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           1.44e-
05
Time:                        09:11:41   Log-Likelihood:                -23.1
98
No. Observations:                 143   AIC:                             54.
40
Df Residuals:                     139   BIC:                             66.
25
Df Model:                           3
Covariance Type:            nonrobust
============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------
--
const          3.0021      0.467      6.426      0.000       2.078       3.9
26
horsepower     0.3572      0.174      2.054      0.042       0.013       0.7
01
boreratio     -0.5064      0.116     -4.354      0.000      -0.736      -0.2
76
wheelbase      0.0187      0.005      3.966      0.000       0.009       0.0
28
============================================================================
==
Omnibus:                       14.699   Durbin-Watson:                   2.2
61
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               45.6
65
Skew:                          -0.171   Prob(JB):                     1.21e-
10
Kurtosis:                       5.747   Cond. No.                     1.95e+
03
============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.95e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

#CHECKING VIF $VIFi$=1/1−Ri2

In [52]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [53]:

```
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[1])]
    vif['VIF'] = round(vif['VIF'], 2)
    vif = vif.sort_values(by = "VIF", ascending = False)
    return(vif)
```

In [54]:

```
x_train_new = build_model(x_train_lm,y_train)
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.1
65
Model:                            OLS   Adj. R-squared:                  0.1
47
Method:                 Least Squares   F-statistic:                     9.1
55
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           1.44e-
05
Time:                        09:11:48   Log-Likelihood:              -23.1
98
No. Observations:                 143   AIC:                             54.
40
Df Residuals:                     139   BIC:                             66.
25
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const          3.0021      0.467      6.426      0.000       2.078       3.9
26
horsepower     0.3572      0.174      2.054      0.042       0.013       0.7
01
boreratio     -0.5064      0.116     -4.354      0.000      -0.736      -0.2
76
wheelbase      0.0187      0.005      3.966      0.000       0.009       0.0
28
==============================================================================
==
Omnibus:                       14.699   Durbin-Watson:                   2.2
61
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               45.6
65
Skew:                          -0.171   Prob(JB):                     1.21e-
10
Kurtosis:                       5.747   Cond. No.                     1.95e+
03
==============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.95e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [55]:

```
x_train_new = build_model(x_train_new,y_train)
```

```
                            OLS Regression Results
============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.1
65
Model:                            OLS   Adj. R-squared:                  0.1
47
Method:                 Least Squares   F-statistic:                     9.1
55
Date:                Sun, 26 Apr 2020   Prob (F-statistic):           1.44e-
05
Time:                        09:11:49   Log-Likelihood:                -23.1
98
No. Observations:                 143   AIC:                             54.
40
Df Residuals:                     139   BIC:                             66.
25
Df Model:                           3
Covariance Type:            nonrobust
============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------
--
const          3.0021      0.467      6.426      0.000       2.078       3.9
26
horsepower     0.3572      0.174      2.054      0.042       0.013       0.7
01
boreratio     -0.5064      0.116     -4.354      0.000      -0.736      -0.2
76
wheelbase      0.0187      0.005      3.966      0.000       0.009       0.0
28
============================================================================
==
Omnibus:                       14.699   Durbin-Watson:                   2.2
61
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               45.6
65
Skew:                          -0.171   Prob(JB):                     1.21e-
10
Kurtosis:                       5.747   Cond. No.                     1.95e+
03
============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.95e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [56]:

```
checkVIF(x_train_new)
```

Out[56]:

	Features	VIF
0	const	374.60
2	boreratio	1.57
1	horsepower	1.41
3	wheelbase	1.35

In [57]:

```
x_train_new = x_train_new.drop(["wheelbase"], axis = 1)
```

In [58]:

```
xtrain_new = build_model(x_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.0
70
Model:                            OLS   Adj. R-squared:                  0.0
57
Method:                 Least Squares   F-statistic:                     5.3
09
Date:                Sun, 26 Apr 2020   Prob (F-statistic):            0.005
99
Time:                        09:12:00   Log-Likelihood:                -30.8
62
No. Observations:                 143   AIC:                             67.
72
Df Residuals:                     140   BIC:                             76.
61
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const          4.2675      0.359     11.895      0.000       3.558       4.9
77
horsepower     0.4794      0.180      2.665      0.009       0.124       0.8
35
boreratio     -0.3406      0.114     -2.985      0.003      -0.566      -0.1
15
==============================================================================
==
Omnibus:                        5.235   Durbin-Watson:                   2.1
61
Prob(Omnibus):                  0.073   Jarque-Bera (JB):                7.5
92
Skew:                           0.049   Prob(JB):                       0.02
25
Kurtosis:                       4.124   Cond. No.                          5
2.8
==============================================================================
==

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

In [59]:

```
checkVIF(x_train_new)
```

Out[59]:

	Features	VIF
0	const	199.87
1	horsepower	1.37
2	boreratio	1.37

Residual analysis of a model

In [60]:

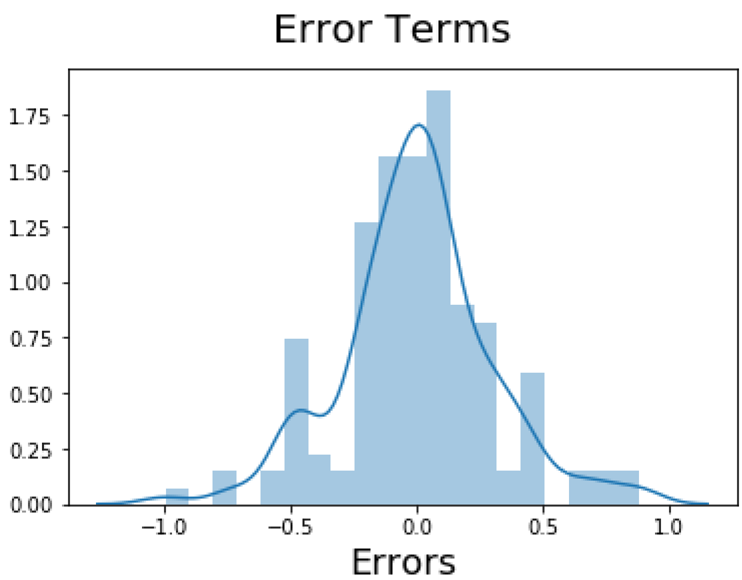
```
lm = sm.OLS(y_train,x_train_new).fit()
y_train_price = lm.predict(x_train_new)
```

In [61]:

```
fig = plt.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)
plt.xlabel('Errors', fontsize = 18)
```

Out[61]:

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



MODEL EVALUATION

In [62]:

```
num_vars = ['wheelbase', 'curbweight', 'enginesize','carlength','carwidth','price']
x_train[num_vars] = scaler.transform(x_train[num_vars])
```

In [63]:

```
y_test = x_train.pop('price')
x_test = x_train
```

In [64]:

```
x_train_new = x_train_new.drop('const',axis=1)
x_test_new = x_test[x_train_new.columns]
x_test_new = sm.add_constant(x_test_new)
```

In [65]:

```
y_pred = lm.predict(x_test_new)
```

In [66]:

```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[66]:

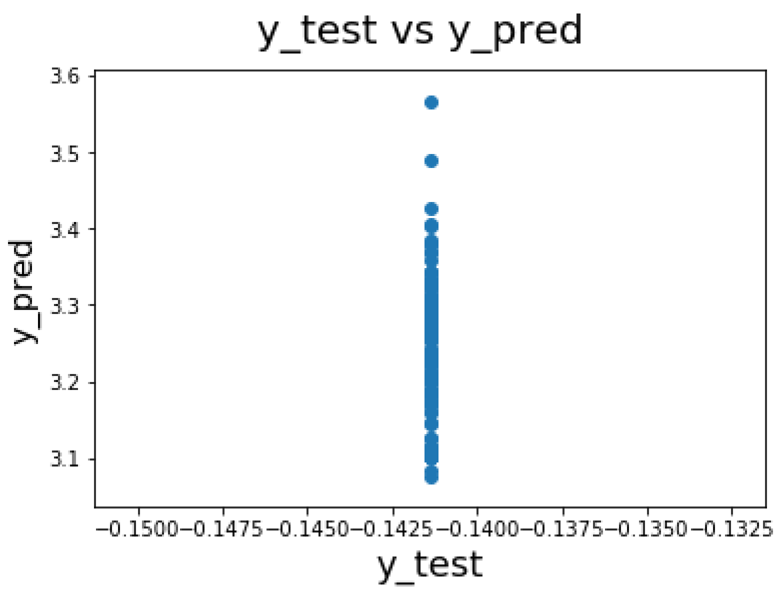
```
-326434617006.14984
```

In [67]:

```
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[67]:

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



In [68]:

```
print(lm.summary())
```

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                 stroke   R-squared:                       0.0
70
Model:                            OLS   Adj. R-squared:                  0.0
57
Method:                 Least Squares   F-statistic:                     5.3
09
Date:                Sun, 26 Apr 2020   Prob (F-statistic):            0.005
99
Time:                        09:12:45   Log-Likelihood:                -30.8
62
No. Observations:                 143   AIC:                             67.
72
Df Residuals:                     140   BIC:                             76.
61
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const          4.2675      0.359     11.895      0.000       3.558       4.9
77
horsepower     0.4794      0.180      2.665      0.009       0.124       0.8
35
boreratio     -0.3406      0.114     -2.985      0.003      -0.566      -0.1
15
==============================================================================
==
Omnibus:                        5.235   Durbin-Watson:                   2.1
61
Prob(Omnibus):                  0.073   Jarque-Bera (JB):                7.5
92
Skew:                           0.049   Prob(JB):                       0.02
25
Kurtosis:                       4.124   Cond. No.                         5
2.8
==============================================================================
==

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

In [ ]:

In [ ]:

```

```