In [1]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

Step 1 : Reading and Understanding the data

In [2]:

```
master_df = pd.read_csv("CarPrice_Assignment.csv")
master_df.set_index('car_ID', inplace = True)
master_df.rename(columns = {"symboling" : "Insuranceriskfactor"}, inplace = True)
```

In [3]:

```
master_df.head()
```

Out[3]:

	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewh
car_ID							
1	3	alfa-romero giulia	gas	std	two	convertible	
2	3	alfa-romero stelvio	gas	std	two	convertible	
3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	
4	2	audi 100 ls	gas	std	four	sedan	
5	2	audi 100ls	gas	std	four	sedan	

5 rows × 25 columns

In [4]:

```
master_df.dtypes
```

Out[4]:

```
Insuranceriskfactor      int64
CarName                 object
fueltype                object
aspiration              object
doornumber              object
carbody                 object
drivewheel              object
enginelocation          object
wheelbase              float64
carlength              float64
carwidth               float64
carheight              float64
curbweight               int64
enginetype              object
cylindernumber          object
enginesize               int64
fuelsystem              object
boreratio              float64
stroke                 float64
compressionratio       float64
horsepower               int64
peakrpm                  int64
citympg                  int64
highwaympg               int64
price                  float64
dtype: object
```

In [5]:

```
master_df.describe()
```

Out[5]:

	Insuranceriskfactor	wheelbase	carlength	carwidth	carheight	curbweight	en(
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326

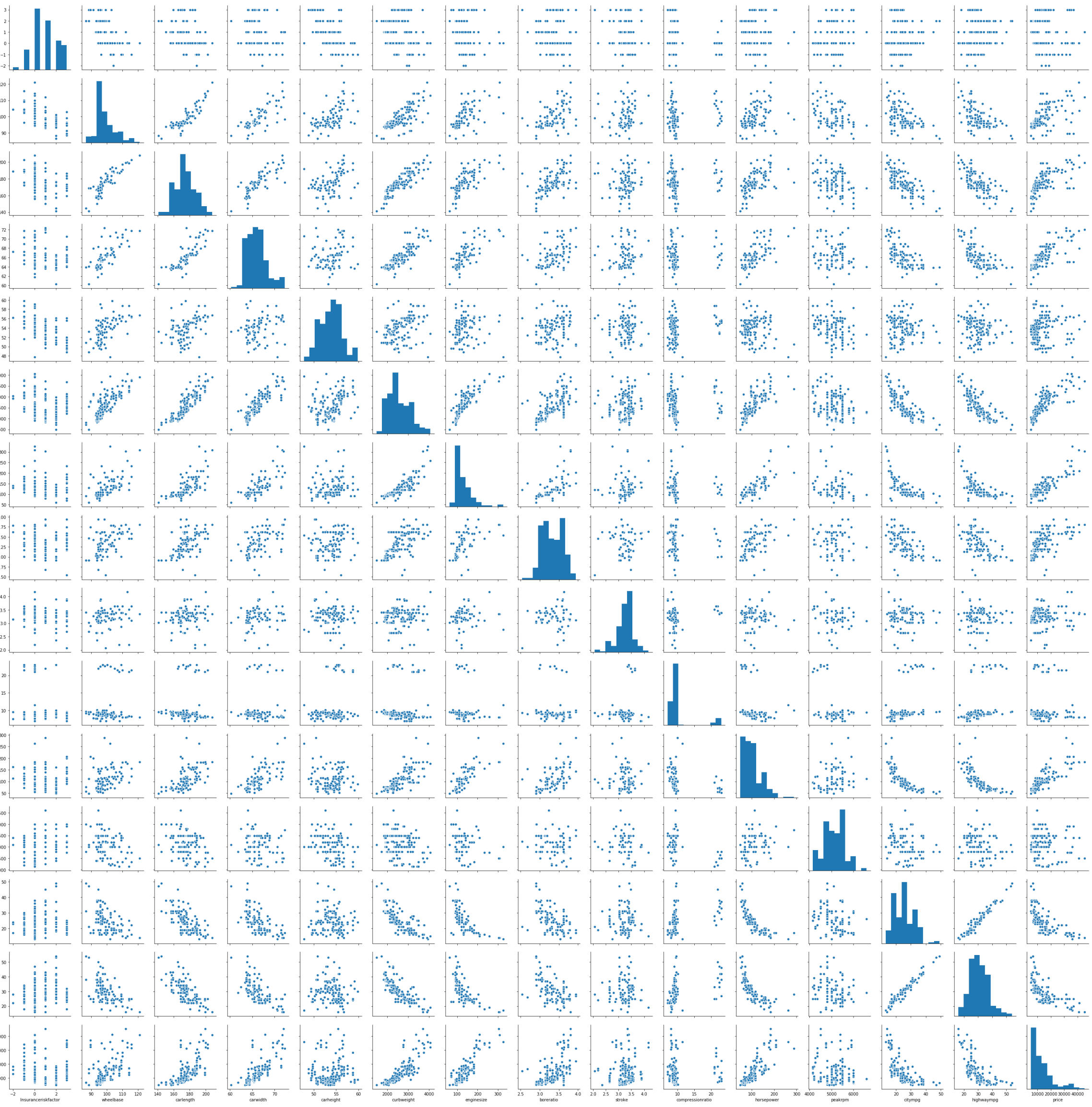
Step 2 : Visualizing the data

Visualizing the Numerical columns

In [69]:

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

numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
num_columns_list = list(master_df.select_dtypes(include=numerics).columns)

plt.figure(figsize = (20,12))
sns.pairplot(master_df[num_columns_list])
plt.show()
```

```
<Figure size 1440x864 with 0 Axes>
```



Visualizing the categorical variables

Restructuring CarName to have on ly the company name

In [7]:

```
def strip_car_model(car_name):
    company_name = car_name.split()
    return company_name[0]

master_df.CarName = master_df.CarName.apply(strip_car_model)
master_df.head()
```

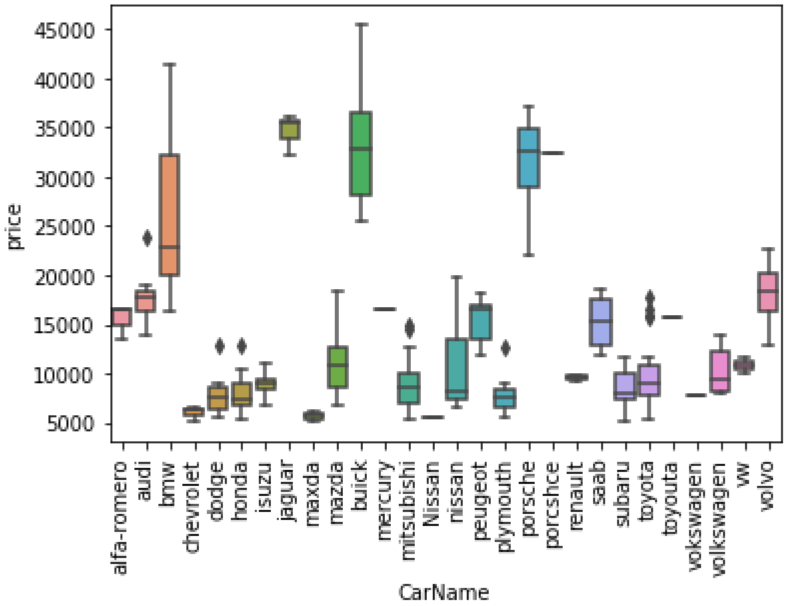
Out[7]:

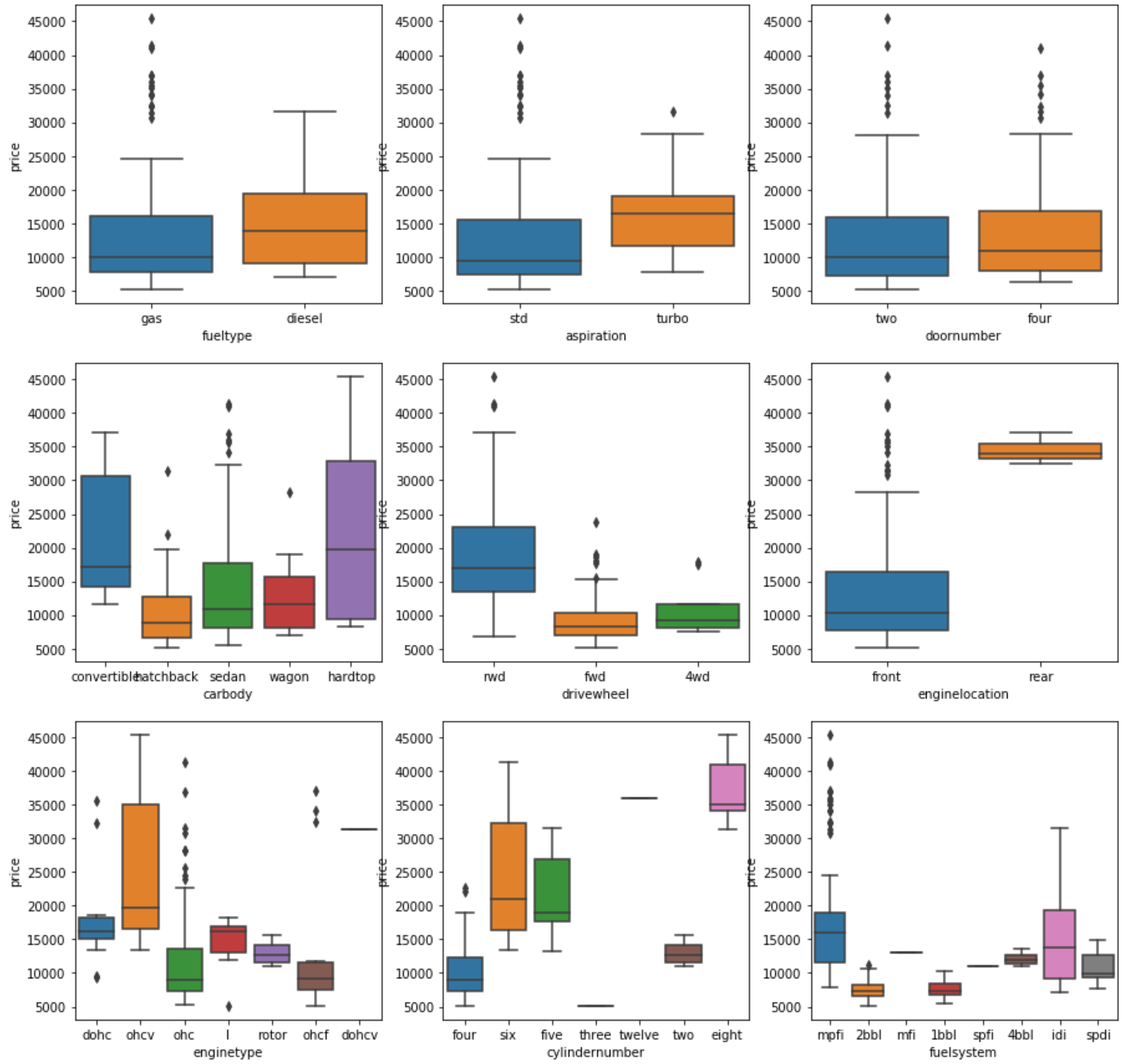
	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewhe
car_ID							
1	3	alfa-romero	gas	std	two	convertible	rw
2	3	alfa-romero	gas	std	two	convertible	rw
3	1	alfa-romero	gas	std	two	hatchback	rw
4	2	audi	gas	std	four	sedan	fw
5	2	audi	gas	std	four	sedan	4w

5 rows × 25 columns

In [8]:

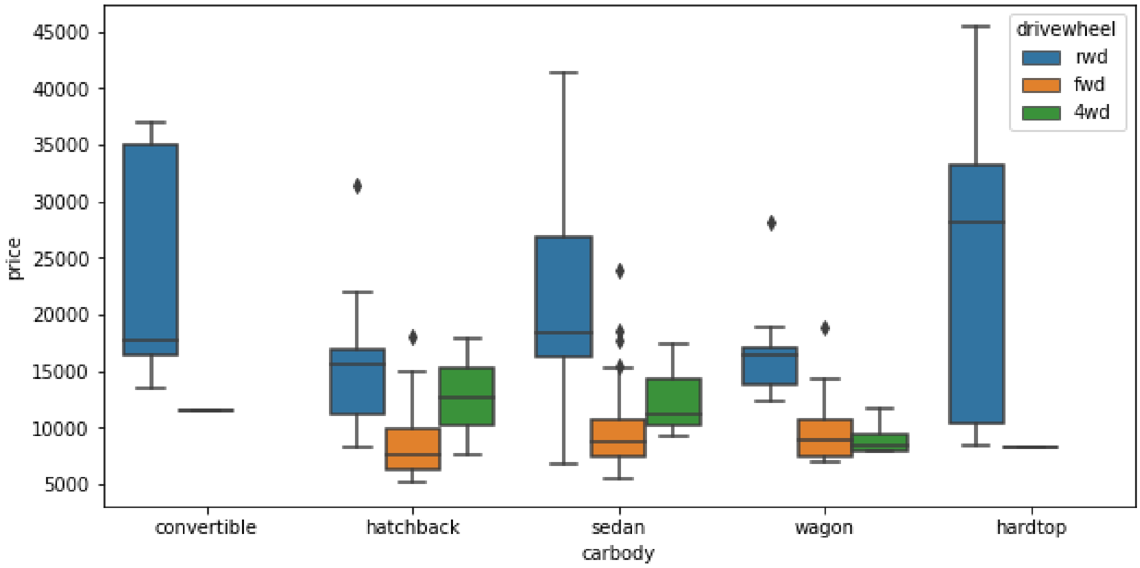
```
non_num_columns_list = list(master_df.select_dtypes(exclude=numerics).columns)

## Visualize the CarName first
sns.boxplot(x= 'CarName', y = 'price', data = master_df)
plt.xticks(rotation='vertical')
plt.show()

subplot_cnt = 1
plt.figure(figsize = (15,15))

for each_cat_var in non_num_columns_list:
    if each_cat_var != 'CarName':
        plt.subplot(3,3,subplot_cnt)
        sns.boxplot(x= each_cat_var, y = 'price', data = master_df)
        subplot_cnt = subplot_cnt+1

plt.show()
```

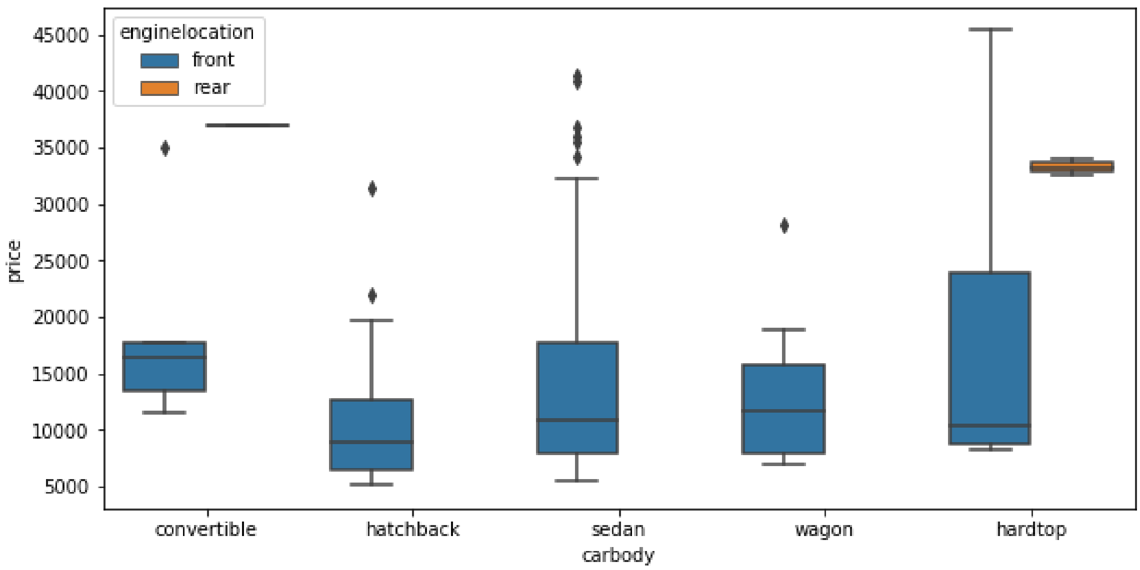

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
alfa-romero
audi
bmw
chevrolet
dodge
honda
isuzu
jaguar
maxda
mazda
buick
mercury
mitsubishi
Nissan
nissan
peugeot
plymouth
porsche
porcshce
renault
saab
subaru
toyota
toyouta
vokswagen
volkswagen
vw
volvo
CarName


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
gas
diesel
fueltype
std
turbo
aspiration
two
four
doornumber
convertible
hatchback
sedan
wagon
hardtop
carbody
rwd
fwd
4wd
drivewheel
front
rear
enginelocation
dohc
ohcv
ohc
l
rotor
ohcf
dohcv
enginetype
four
six
five
three
twelve
two
eight
cylindernumber
mpfi
2bbl
mfi
1bbl
spfi
4bbl
idi
spdi
fuelsystem

In [9]:

```
## Visualize the carbody and driveweheel impact on the price
plt.figure(figsize = (10, 5))
sns.boxplot(x = 'carbody', y = 'price', hue = 'drivewheel', data = master_df)
plt.show()

## Visualize the carbody and enginelocation impact on the price
plt.figure(figsize = (10, 5))
sns.boxplot(x = 'carbody', y = 'price', hue = 'enginelocation', data = master_df)
plt.show()
```





Step 3 : Data Preparation

In [10]:

```
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder

## Create a copy of df
master_df_le = master_df.copy()

## Apply lable enabler on all the categorical variables
le = preprocessing.LabelEncoder()

for each_item in non_num_columns_list:
    le.fit(master_df_le[each_item])
    master_df_le[each_item] = le.transform(master_df_le[each_item])

master_df_le.head()
```

Out[10]:

	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
car_ID							
1	3	1	1	0	1	0	2
2	3	1	1	0	1	0	2
3	1	1	1	0	1	2	2
4	2	2	1	0	0	3	1
5	2	2	1	0	0	3	0

5 rows × 25 columns

Step 4: Splitting the training and test data

In [11]:

```
import numpy as np
from sklearn.model_selection import train_test_split

np.random.seed(0)
master_df_train, master_df_test = train_test_split(master_df_le, train_size = 0.7, test
_size = 0.3, random_state = 100)
master_df_train.head()
```

Out[11]:

	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
car_ID							
123	1	16	1	0	0	3	1
126	3	18	1	0	1	2	2
167	1	22	1	0	1	2	2
2	3	1	1	0	1	0	2
200	-1	26	1	1	0	4	2

5 rows × 25 columns

Re-sclaing the parameters

In [12]:

```
## Scale and transform all the numerical values
## especially the car dimentions, wheelbase and price are on very hig scale in comparis
on to the
## let us use minmaxmethod - as it helps normalize the outliers

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
master_df_train[num_columns_list] = scaler.fit_transform(master_df_train[num_columns_li
st])
master_df_train[non_num_columns_list] = scaler.fit_transform(master_df_train[non_num_co
lumns_list])
master_df_train.head()
```

Out[12]:

	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
car_ID							
123	0.6	0.592593	1.0	0.0	0.0	0.75	0.5
126	1.0	0.666667	1.0	0.0	1.0	0.50	1.0
167	0.6	0.814815	1.0	0.0	1.0	0.50	1.0
2	1.0	0.037037	1.0	0.0	1.0	0.00	1.0
200	0.2	0.962963	1.0	1.0	0.0	1.00	1.0

5 rows × 25 columns

In [13]:

```
master_df_train.describe()
```

Out[13]:

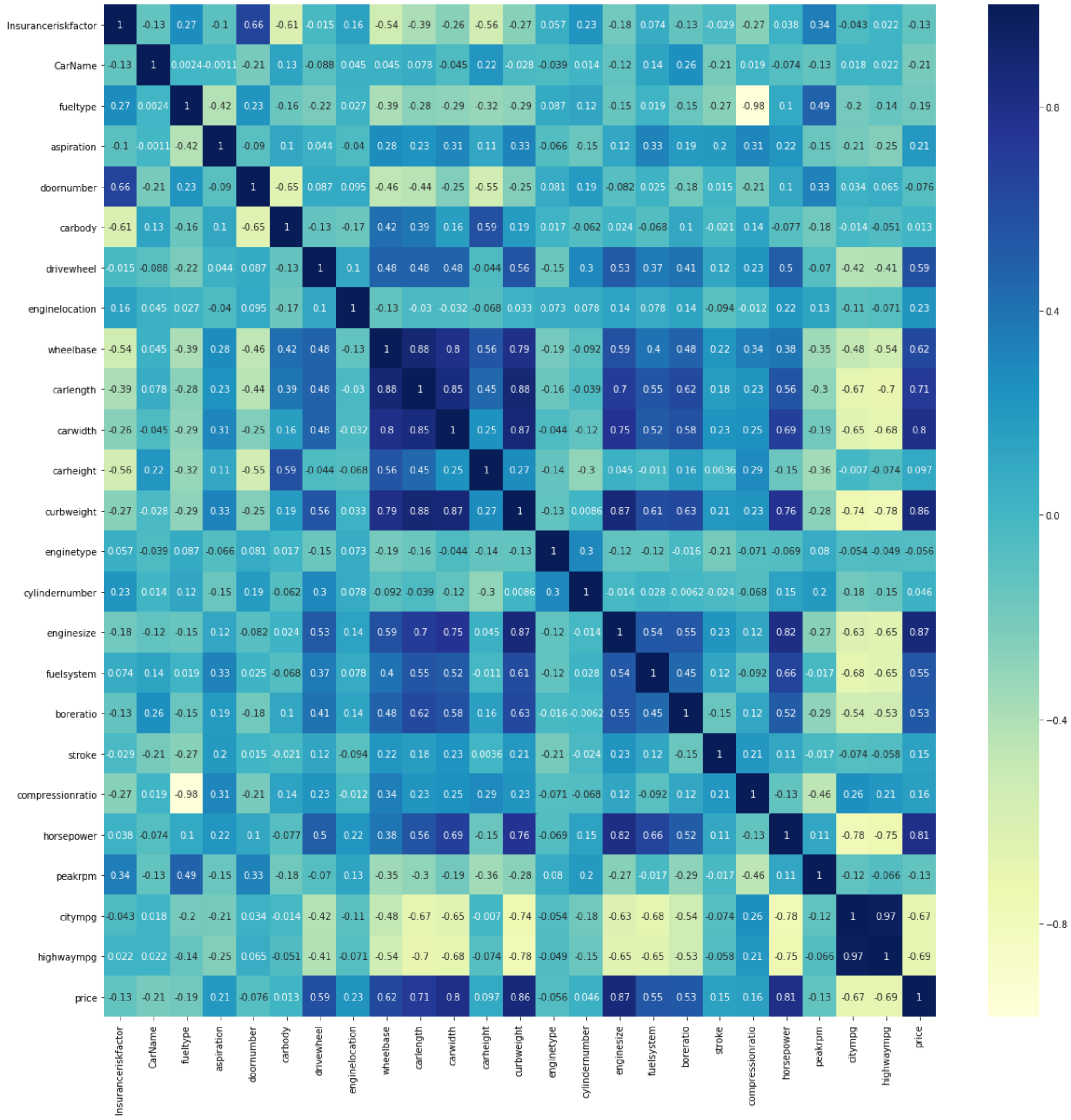
	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	driv
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143
mean	0.559441	0.517742	0.909091	0.181818	0.440559	0.666084	0
std	0.239200	0.276478	0.288490	0.387050	0.498199	0.209678	0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	0.400000	0.259259	1.000000	0.000000	0.000000	0.500000	0
50%	0.600000	0.518519	1.000000	0.000000	0.000000	0.750000	0
75%	0.600000	0.777778	1.000000	0.000000	1.000000	0.750000	1
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1

8 rows × 25 columns

Calculating the corelation between different parameters of the dataframe.

In [14]:

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


Insuranceriskfactor
CarName
fueltype
aspiration
doornumber
carbody
drivewheel
enginelocation
wheelbase
carlength
carwidth
carheight
curbweight
enginetype
cylindernumber
enginesize
fuelsystem
boreratio
stroke
compressionratio
horsepower
peakrpm
citympg
highwaympg
price
0.8
0.4
0.0
−0.4
−0.8

Step 5: Building a linear model

In [15]:

```
## Prepare the X and y train data

cols_list = list(master_df_train.columns)
cols_list.remove('price')

y_train = master_df_train['price']
X_train = master_df_train[cols_list]
```

In [16]:

```
y_train.head()
```

Out[16]:

```
car_ID
123    0.068818
126    0.466890
167    0.122110
2      0.314446
200    0.382131
Name: price, dtype: float64
```

In [17]:

```
X_train.head()
```

Out[17]:

	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
car_ID							
123	0.6	0.592593	1.0	0.0	0.0	0.75	0.5
126	1.0	0.666667	1.0	0.0	1.0	0.50	1.0
167	0.6	0.814815	1.0	0.0	1.0	0.50	1.0
2	1.0	0.037037	1.0	0.0	1.0	0.00	1.0
200	0.2	0.962963	1.0	1.0	0.0	1.00	1.0

5 rows × 24 columns

Model 1 : Adding all the parameters (predictors) to build the model

In [18]:

```
import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train)
lr_1 = sm.OLS(y_train, X_train_lm).fit()
lr_1.summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.886
Model:	OLS	**Adj. R-squared:**	0.863
Method:	Least Squares	**F-statistic:**	38.15
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	1.45e-44
Time:	10:34:06	**Log-Likelihood:**	172.11
No. Observations:	143	**AIC:**	-294.2
Df Residuals:	118	**BIC:**	-220.2
Df Model:	24		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.2899	0.302	-0.959	0.340	-0.889	0.309
Insuranceriskfactor	0.0232	0.049	0.475	0.635	-0.073	0.120
CarName	-0.1349	0.031	-4.397	0.000	-0.196	-0.074
fueltype	0.1255	0.253	0.496	0.621	-0.376	0.627
aspiration	0.0115	0.031	0.370	0.712	-0.050	0.073
doornumber	-0.0134	0.023	-0.597	0.552	-0.058	0.031
carbody	-0.0628	0.055	-1.150	0.252	-0.171	0.045
drivewheel	0.0525	0.038	1.376	0.172	-0.023	0.128
enginelocation	0.3261	0.095	3.449	0.001	0.139	0.513
wheelbase	0.0696	0.111	0.627	0.532	-0.150	0.290
carlength	-0.0765	0.121	-0.631	0.529	-0.317	0.164
carwidth	0.2379	0.126	1.886	0.062	-0.012	0.488
carheight	0.0897	0.055	1.638	0.104	-0.019	0.198
curbweight	0.2757	0.146	1.893	0.061	-0.013	0.564
enginetype	0.0280	0.049	0.577	0.565	-0.068	0.124
cylindernumber	0.0449	0.072	0.624	0.534	-0.098	0.187
enginesize	0.4803	0.160	3.008	0.003	0.164	0.796
fuelsystem	0.0302	0.035	0.872	0.385	-0.038	0.099
boreratio	-0.0124	0.054	-0.229	0.819	-0.119	0.095
stroke	-0.1081	0.061	-1.774	0.079	-0.229	0.013
compressionratio	0.1755	0.292	0.602	0.548	-0.402	0.753
horsepower	0.1702	0.157	1.087	0.279	-0.140	0.480
peakrpm	0.0537	0.056	0.968	0.335	-0.056	0.164
citympg	-0.0530	0.211	-0.251	0.802	-0.471	0.365
highwaympg	0.0851	0.199	0.428	0.669	-0.308	0.479

Omnibus:	44.657	**Durbin-Watson:**	1.814

Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	245.736
Skew:	0.933	**Prob(JB):**	4.36e-54
Kurtosis:	9.145	**Cond. No.**	184.

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

In [19]:

```
## Calculate the VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

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

Out[19]:

	Features	VIF
2	fueltype	157.95
23	highwaympg	155.34
22	citympg	152.24
9	carlength	104.38
12	curbweight	94.01
10	carwidth	83.98
8	wheelbase	56.21
15	enginesize	45.88
20	horsepower	43.08
19	compressionratio	32.77
5	carbody	32.47
11	carheight	20.26
18	stroke	20.26
0	Insuranceriskfactor	18.48
17	boreratio	17.16
14	cylindernumber	16.77
6	drivewheel	16.13
13	enginetype	14.34
21	peakrpm	13.53
16	fuelsystem	9.69
1	CarName	6.78
4	doornumber	4.99
3	aspiration	2.81
7	enginelocation	1.40

Model 2 - Dropping the fueltype predictor as it has the highest P value (insignificane)

In [20]:

```
X = X_train.drop('fueltype',1)
X_train_lm = sm.add_constant(X)
lr_2 = sm.OLS(y_train, X_train_lm).fit()
lr_2.summary()
```

Out[20]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.886
Model:	OLS	**Adj. R-squared:**	0.863
Method:	Least Squares	**F-statistic:**	40.05
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	2.54e-45
Time:	10:34:06	**Log-Likelihood:**	171.96
No. Observations:	143	**AIC:**	-295.9
Df Residuals:	119	**BIC:**	-224.8
Df Model:	23		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1494	0.105	-1.423	0.157	-0.357	0.058
Insuranceriskfactor	0.0202	0.048	0.419	0.676	-0.075	0.116
CarName	-0.1302	0.029	-4.481	0.000	-0.188	-0.073
aspiration	0.0035	0.026	0.132	0.895	-0.049	0.056
doornumber	-0.0131	0.022	-0.585	0.560	-0.058	0.031
carbody	-0.0622	0.054	-1.144	0.255	-0.170	0.045
drivewheel	0.0548	0.038	1.452	0.149	-0.020	0.129
enginelocation	0.3239	0.094	3.440	0.001	0.137	0.510
wheelbase	0.0581	0.108	0.536	0.593	-0.156	0.272
carlength	-0.0686	0.120	-0.573	0.568	-0.306	0.169
carwidth	0.2515	0.123	2.049	0.043	0.008	0.495
carheight	0.0887	0.055	1.626	0.107	-0.019	0.197
curbweight	0.2624	0.143	1.839	0.068	-0.020	0.545
enginetype	0.0253	0.048	0.526	0.600	-0.070	0.121
cylindernumber	0.0550	0.069	0.800	0.425	-0.081	0.191
enginesize	0.4927	0.157	3.135	0.002	0.181	0.804
fuelsystem	0.0264	0.034	0.785	0.434	-0.040	0.093
boreratio	-0.0194	0.052	-0.372	0.710	-0.122	0.084
stroke	-0.1207	0.055	-2.186	0.031	-0.230	-0.011
compressionratio	0.0330	0.049	0.679	0.498	-0.063	0.129
horsepower	0.1762	0.156	1.132	0.260	-0.132	0.484
peakrpm	0.0599	0.054	1.112	0.268	-0.047	0.167
citympg	-0.0492	0.210	-0.234	0.816	-0.466	0.367
highwaympg	0.0885	0.198	0.447	0.656	-0.303	0.480

Omnibus:	40.492	**Durbin-Watson:**	1.821
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	199.345

Skew:	0.859	**Prob(JB):**	5.16e-44
Kurtosis:	8.523	**Cond. No.**	105.

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

In [21]:

```
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)
vif
```

Out[21]:

	Features	VIF
22	highwaympg	153.12
21	citympg	143.50
8	carlength	98.53
11	curbweight	92.30
9	carwidth	83.72
7	wheelbase	55.66
14	enginesize	45.58
19	horsepower	43.02
4	carbody	31.89
17	stroke	20.17
10	carheight	19.69
16	boreratio	17.01
0	Insuranceriskfactor	16.72
13	cylindernumber	16.19
5	drivewheel	15.74
12	enginetype	12.85
20	peakrpm	11.61
15	fuelsystem	9.69
1	CarName	6.44
3	doornumber	4.97
18	compressionratio	4.17
2	aspiration	2.79
6	enginelocation	1.39

In [ ]:

```

```

Model 3 - Dropping the fueltype predictor as it has the highest P value (insignificane)

In [22]:

```
X = X.drop('Insuranceriskfactor',1)
X_train_lm = sm.add_constant(X)
lr_3 = sm.OLS(y_train, X_train_lm).fit()
lr_3.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.885
Model:	OLS	**Adj. R-squared:**	0.864
Method:	Least Squares	**F-statistic:**	42.15
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	4.20e-46
Time:	10:34:07	**Log-Likelihood:**	171.86
No. Observations:	143	**AIC:**	-297.7
Df Residuals:	120	**BIC:**	-229.6
Df Model:	22		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1332	0.097	-1.369	0.174	-0.326	0.059
CarName	-0.1312	0.029	-4.548	0.000	-0.188	-0.074
aspiration	0.0034	0.026	0.130	0.897	-0.049	0.055
doornumber	-0.0094	0.021	-0.458	0.648	-0.050	0.031
carbody	-0.0656	0.054	-1.222	0.224	-0.172	0.041
drivewheel	0.0544	0.038	1.447	0.150	-0.020	0.129
enginelocation	0.3287	0.093	3.531	0.001	0.144	0.513
wheelbase	0.0405	0.100	0.407	0.685	-0.156	0.238
carlength	-0.0670	0.119	-0.561	0.576	-0.303	0.169
carwidth	0.2628	0.119	2.201	0.030	0.026	0.499
carheight	0.0877	0.054	1.615	0.109	-0.020	0.195
curbweight	0.2681	0.142	1.894	0.061	-0.012	0.548
enginetype	0.0210	0.047	0.448	0.655	-0.072	0.114
cylindernumber	0.0610	0.067	0.909	0.365	-0.072	0.194
enginesize	0.4882	0.156	3.124	0.002	0.179	0.798
fuelsystem	0.0287	0.033	0.868	0.387	-0.037	0.094
boreratio	-0.0185	0.052	-0.357	0.722	-0.121	0.084
stroke	-0.1209	0.055	-2.197	0.030	-0.230	-0.012
compressionratio	0.0323	0.048	0.669	0.505	-0.063	0.128
horsepower	0.1623	0.152	1.071	0.286	-0.138	0.462
peakrpm	0.0605	0.054	1.127	0.262	-0.046	0.167
citympg	-0.0592	0.208	-0.284	0.777	-0.472	0.353
highwaympg	0.0934	0.197	0.474	0.636	-0.296	0.483

Omnibus:	38.950	**Durbin-Watson:**	1.816
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	181.348
Skew:	0.838	**Prob(JB):**	4.18e-40

Kurtosis:	8.256	**Cond. No.**	101.

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

In [23]:

```
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)
vif
```

Out[23]:

	Features	VIF
21	highwaympg	151.28
20	citympg	143.39
7	carlength	97.12
10	curbweight	88.96
8	carwidth	79.46
6	wheelbase	45.55
13	enginesize	45.40
18	horsepower	40.86
3	carbody	31.54
16	stroke	20.01
9	carheight	19.67
15	boreratio	16.75
4	drivewheel	15.73
12	cylindernumber	15.11
11	enginetype	12.76
19	peakrpm	11.29
14	fuelsystem	9.29
0	CarName	6.43
17	compressionratio	4.00
2	doornumber	3.97
1	aspiration	2.78
5	enginelocation	1.37

In [ ]:

```
```

Model 4 - Dropping the carlength predictor as it has the highest P value and VIF (insignificane)

In [24]:

```
X = X.drop('carlength',1)
X_train_lm = sm.add_constant(X)
lr_4 = sm.OLS(y_train, X_train_lm).fit()
lr_4.summary()
```

Out[24]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.885
Model:	OLS	**Adj. R-squared:**	0.865
Method:	Least Squares	**F-statistic:**	44.40
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	7.24e-47
Time:	10:34:07	**Log-Likelihood:**	171.67
No. Observations:	143	**AIC:**	-299.3
Df Residuals:	121	**BIC:**	-234.2
Df Model:	21		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1460	0.094	-1.548	0.124	-0.333	0.041
CarName	-0.1313	0.029	-4.566	0.000	-0.188	-0.074
aspiration	0.0057	0.026	0.222	0.825	-0.045	0.057
doornumber	-0.0081	0.020	-0.397	0.692	-0.048	0.032
carbody	-0.0752	0.051	-1.484	0.140	-0.176	0.025
drivewheel	0.0536	0.037	1.430	0.155	-0.021	0.128
enginelocation	0.3252	0.093	3.511	0.001	0.142	0.509
wheelbase	0.0294	0.097	0.302	0.763	-0.163	0.222
carwidth	0.2381	0.111	2.152	0.033	0.019	0.457
carheight	0.0835	0.054	1.557	0.122	-0.023	0.190
curbweight	0.2565	0.140	1.837	0.069	-0.020	0.533
enginetype	0.0282	0.045	0.628	0.531	-0.061	0.117
cylindernumber	0.0556	0.066	0.840	0.403	-0.075	0.187
enginesize	0.4787	0.155	3.090	0.002	0.172	0.785
fuelsystem	0.0239	0.032	0.751	0.454	-0.039	0.087
boreratio	-0.0228	0.051	-0.446	0.656	-0.124	0.078
stroke	-0.1203	0.055	-2.192	0.030	-0.229	-0.012
compressionratio	0.0306	0.048	0.637	0.526	-0.065	0.126
horsepower	0.1826	0.147	1.244	0.216	-0.108	0.473
peakrpm	0.0622	0.053	1.163	0.247	-0.044	0.168
citympg	-0.0352	0.203	-0.173	0.863	-0.438	0.367
highwaympg	0.0886	0.196	0.451	0.652	-0.300	0.477

Omnibus:	35.678	**Durbin-Watson:**	1.811
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	153.022
Skew:	0.776	**Prob(JB):**	5.91e-34
Kurtosis:	7.824	**Cond. No.**	96.9

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

In [25]:

```
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)
vif
```

Out[25]:

	Features	VIF
20	highwaympg	150.28
19	citympg	139.60
9	curbweight	84.59
7	carwidth	67.10
12	enginesize	44.83
6	wheelbase	44.36
17	horsepower	38.03
3	carbody	27.64
15	stroke	19.98
8	carheight	18.95
14	boreratio	15.99
4	drivewheel	15.65
11	cylindernumber	14.54
10	enginetype	12.17
18	peakrpm	11.28
13	fuelsystem	8.46
0	CarName	6.42
16	compressionratio	3.99
2	doornumber	3.95
1	aspiration	2.73
5	enginelocation	1.37

In [ ]:

```

```

Model 5 - Dropping the fuelsystem predictor as it has the highest P value (insignificane)

In [26]:

```
X = X.drop('fuelsystem',1)
X_train_lm = sm.add_constant(X)
lr_5 = sm.OLS(y_train, X_train_lm).fit()
lr_5.summary()
```

Out[26]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.885
Model:	OLS	**Adj. R-squared:**	0.866
Method:	Least Squares	**F-statistic:**	46.76
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	1.37e-47
Time:	10:34:07	**Log-Likelihood:**	171.34
No. Observations:	143	**AIC:**	-300.7
Df Residuals:	122	**BIC:**	-238.5
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1310	0.092	-1.424	0.157	-0.313	0.051
CarName	-0.1258	0.028	-4.533	0.000	-0.181	-0.071
aspiration	0.0093	0.025	0.365	0.716	-0.041	0.059
doornumber	-0.0070	0.020	-0.345	0.730	-0.047	0.033
carbody	-0.0794	0.050	-1.580	0.117	-0.179	0.020
drivewheel	0.0537	0.037	1.435	0.154	-0.020	0.128
enginelocation	0.3221	0.092	3.487	0.001	0.139	0.505
wheelbase	0.0407	0.096	0.425	0.672	-0.149	0.231
carwidth	0.2275	0.110	2.076	0.040	0.011	0.444
carheight	0.0814	0.053	1.522	0.131	-0.024	0.187
curbweight	0.2621	0.139	1.882	0.062	-0.014	0.538
enginetype	0.0275	0.045	0.614	0.540	-0.061	0.116
cylindernumber	0.0499	0.066	0.761	0.448	-0.080	0.180
enginesize	0.4754	0.155	3.075	0.003	0.169	0.781
boreratio	-0.0241	0.051	-0.472	0.637	-0.125	0.077
stroke	-0.1192	0.055	-2.177	0.031	-0.228	-0.011
compressionratio	0.0293	0.048	0.610	0.543	-0.066	0.124
horsepower	0.2009	0.144	1.390	0.167	-0.085	0.487
peakrpm	0.0598	0.053	1.122	0.264	-0.046	0.165
citympg	-0.0725	0.197	-0.369	0.713	-0.462	0.317
highwaympg	0.1113	0.193	0.575	0.566	-0.272	0.494

Omnibus:	35.529	**Durbin-Watson:**	1.814
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	147.274
Skew:	0.786	**Prob(JB):**	1.05e-32
Kurtosis:	7.717	**Cond. No.**	92.9

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

In [27]:

```
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)
vif
```

Out[27]:

	Features	VIF
19	highwaympg	144.56
18	citympg	133.13
9	curbweight	83.21
7	carwidth	66.48
12	enginesize	44.80
6	wheelbase	43.65
16	horsepower	37.27
3	carbody	27.48
14	stroke	19.84
8	carheight	18.95
13	boreratio	15.98
4	drivewheel	15.64
11	cylindernumber	14.47
10	enginetype	12.15
17	peakrpm	11.28
0	CarName	5.89
15	compressionratio	3.91
2	doornumber	3.89
1	aspiration	2.61
5	enginelocation	1.37

Model 6 - Dropping the cylindernumber predictor as it has the highest P value (insignificane)

In [28]:

```
X = X.drop('cylindernumber',1)
X_train_lm = sm.add_constant(X)
lr_6 = sm.OLS(y_train, X_train_lm).fit()
lr_6.summary()
```

Out[28]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.884
Model:	OLS	**Adj. R-squared:**	0.866
Method:	Least Squares	**F-statistic:**	49.36
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	2.55e-48
Time:	10:34:07	**Log-Likelihood:**	171.00
No. Observations:	143	**AIC:**	-302.0
Df Residuals:	123	**BIC:**	-242.7
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1165	0.090	-1.297	0.197	-0.294	0.061
CarName	-0.1239	0.028	-4.491	0.000	-0.179	-0.069
aspiration	0.0028	0.024	0.118	0.906	-0.044	0.050
doornumber	-0.0067	0.020	-0.330	0.742	-0.047	0.033
carbody	-0.0746	0.050	-1.499	0.136	-0.173	0.024
drivewheel	0.0609	0.036	1.688	0.094	-0.011	0.132
enginelocation	0.3205	0.092	3.477	0.001	0.138	0.503
wheelbase	0.0556	0.094	0.593	0.554	-0.130	0.241
carwidth	0.1912	0.098	1.942	0.054	-0.004	0.386
carheight	0.0688	0.051	1.356	0.178	-0.032	0.169
curbweight	0.2768	0.138	2.011	0.046	0.004	0.549
enginetype	0.0420	0.041	1.035	0.303	-0.038	0.122
enginesize	0.4369	0.146	2.997	0.003	0.148	0.725
boreratio	-0.0255	0.051	-0.501	0.617	-0.126	0.075
stroke	-0.1112	0.054	-2.073	0.040	-0.217	-0.005
compressionratio	0.0365	0.047	0.778	0.438	-0.056	0.130
horsepower	0.2358	0.137	1.724	0.087	-0.035	0.507
peakrpm	0.0570	0.053	1.074	0.285	-0.048	0.162
citympg	-0.0622	0.196	-0.317	0.751	-0.450	0.326
highwaympg	0.0939	0.192	0.490	0.625	-0.286	0.473

Omnibus:	37.735	**Durbin-Watson:**	1.819
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	155.443
Skew:	0.851	**Prob(JB):**	1.76e-34
Kurtosis:	7.816	**Cond. No.**	90.0

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

In [29]:

```
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)
vif
```

Out[29]:

	Features	VIF
18	highwaympg	143.64
17	citympg	131.82
9	curbweight	78.98
7	carwidth	54.81
6	wheelbase	42.19
11	enginesize	39.92
15	horsepower	33.82
3	carbody	26.73
13	stroke	18.45
8	carheight	17.50
12	boreratio	15.97
4	drivewheel	14.30
16	peakrpm	11.28
10	enginetype	8.93
0	CarName	5.80
2	doornumber	3.87
14	compressionratio	3.85
1	aspiration	2.36
5	enginelocation	1.36

Model 7 - Dropping the citympg predictor as it has the highest P value (insignificane) and VIF

In [30]:

```
X = X.drop('citympg',1)
X_train_lm = sm.add_constant(X)
lr_7 = sm.OLS(y_train, X_train_lm).fit()
lr_7.summary()
```

Out[30]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.884
Model:	OLS	**Adj. R-squared:**	0.867
Method:	Least Squares	**F-statistic:**	52.47
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	3.64e-49
Time:	10:34:07	**Log-Likelihood:**	170.94
No. Observations:	143	**AIC:**	-303.9
Df Residuals:	124	**BIC:**	-247.6
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1208	0.089	-1.365	0.175	-0.296	0.054
CarName	-0.1242	0.027	-4.520	0.000	-0.179	-0.070
aspiration	0.0009	0.023	0.040	0.968	-0.045	0.046
doornumber	-0.0075	0.020	-0.377	0.707	-0.047	0.032
carbody	-0.0748	0.050	-1.508	0.134	-0.173	0.023
drivewheel	0.0619	0.036	1.730	0.086	-0.009	0.133
enginelocation	0.3226	0.092	3.521	0.001	0.141	0.504
wheelbase	0.0526	0.093	0.566	0.572	-0.131	0.236
carwidth	0.1921	0.098	1.958	0.052	-0.002	0.386
carheight	0.0677	0.050	1.342	0.182	-0.032	0.168
curbweight	0.2799	0.137	2.046	0.043	0.009	0.551
enginetype	0.0428	0.040	1.061	0.291	-0.037	0.123
enginesize	0.4219	0.137	3.070	0.003	0.150	0.694
boreratio	-0.0220	0.049	-0.445	0.657	-0.120	0.076
stroke	-0.1057	0.051	-2.090	0.039	-0.206	-0.006
compressionratio	0.0344	0.046	0.742	0.459	-0.057	0.126
horsepower	0.2502	0.129	1.945	0.054	-0.004	0.505
peakrpm	0.0554	0.053	1.052	0.295	-0.049	0.160
highwaympg	0.0396	0.087	0.458	0.648	-0.132	0.211

Omnibus:	38.156	**Durbin-Watson:**	1.819
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	157.707
Skew:	0.862	**Prob(JB):**	5.68e-35
Kurtosis:	7.847	**Cond. No.**	58.2

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

In [31]:

```
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)
vif
```

Out[31]:

	Features	VIF
9	curbweight	78.96
7	carwidth	54.79
6	wheelbase	41.86
11	enginesize	35.81
15	horsepower	29.88
3	carbody	26.69
8	carheight	17.34
13	stroke	16.85
12	boreratio	15.44
17	highwaympg	15.24
4	drivewheel	14.25
16	peakrpm	11.00
10	enginetype	8.93
0	CarName	5.77
14	compressionratio	3.82
2	doornumber	3.74
1	aspiration	2.18
5	enginelocation	1.36

Model 8 - Dropping the aspiration predictor as it has the highest P value (insignificane) and VIF

In [32]:

```
X = X.drop('aspiration',1)
X_train_lm = sm.add_constant(X)
lr_8 = sm.OLS(y_train, X_train_lm).fit()
lr_8.summary()
```

Out[32]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.884
Model:	OLS	**Adj. R-squared:**	0.868
Method:	Least Squares	**F-statistic:**	56.01
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	4.78e-50
Time:	10:34:07	**Log-Likelihood:**	170.94
No. Observations:	143	**AIC:**	-305.9
Df Residuals:	125	**BIC:**	-252.6
Df Model:	17		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.1203	0.087	-1.378	0.171	-0.293	0.053
CarName	-0.1243	0.027	-4.564	0.000	-0.178	-0.070
doornumber	-0.0075	0.020	-0.377	0.707	-0.047	0.032
carbody	-0.0748	0.049	-1.514	0.133	-0.173	0.023
drivewheel	0.0615	0.034	1.804	0.074	-0.006	0.129
enginelocation	0.3227	0.091	3.536	0.001	0.142	0.503
wheelbase	0.0530	0.092	0.578	0.565	-0.129	0.235
carwidth	0.1918	0.097	1.969	0.051	-0.001	0.385
carheight	0.0675	0.050	1.352	0.179	-0.031	0.166
curbweight	0.2806	0.135	2.078	0.040	0.013	0.548
enginetype	0.0427	0.040	1.065	0.289	-0.037	0.122
enginesize	0.4193	0.120	3.485	0.001	0.181	0.657
boreratio	-0.0220	0.049	-0.446	0.656	-0.119	0.075
stroke	-0.1053	0.049	-2.132	0.035	-0.203	-0.008
compressionratio	0.0350	0.044	0.802	0.424	-0.051	0.121
horsepower	0.2520	0.120	2.107	0.037	0.015	0.489
peakrpm	0.0549	0.051	1.080	0.282	-0.046	0.155
highwaympg	0.0392	0.086	0.458	0.648	-0.130	0.209

Omnibus:	38.092	**Durbin-Watson:**	1.818
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	157.385
Skew:	0.860	**Prob(JB):**	6.67e-35
Kurtosis:	7.843	**Cond. No.**	54.8

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

In [33]:

```
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)
vif
```

Out[33]:

	Features	VIF
8	curbweight	75.77
6	carwidth	54.53
5	wheelbase	41.35
10	enginesize	27.51
2	carbody	26.67
14	horsepower	26.22
7	carheight	17.19
12	stroke	15.80
11	boreratio	15.36
16	highwaympg	15.24
3	drivewheel	13.18
15	peakrpm	10.50
9	enginetype	8.93
0	CarName	5.74
1	doornumber	3.71
13	compressionratio	3.48
4	enginelocation	1.36

Model 9 - Dropping the highwaympg predictor as it has the highest P value (insignificane) and VIF

In [34]:

```
X = X.drop('highwaympg',1)
X_train_lm = sm.add_constant(X)
lr_9 = sm.OLS(y_train, X_train_lm).fit()
lr_9.summary()
```

Out[34]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.884
Model:	OLS	**Adj. R-squared:**	0.869
Method:	Least Squares	**F-statistic:**	59.87
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	6.73e-51
Time:	10:34:07	**Log-Likelihood:**	170.82
No. Observations:	143	**AIC:**	-307.6
Df Residuals:	126	**BIC:**	-257.3
Df Model:	16		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0922	0.062	-1.489	0.139	-0.215	0.030
CarName	-0.1241	0.027	-4.572	0.000	-0.178	-0.070
doornumber	-0.0078	0.020	-0.395	0.694	-0.047	0.031
carbody	-0.0714	0.049	-1.467	0.145	-0.168	0.025
drivewheel	0.0625	0.034	1.842	0.068	-0.005	0.130
enginelocation	0.3216	0.091	3.537	0.001	0.142	0.502
wheelbase	0.0470	0.091	0.519	0.605	-0.132	0.226
carwidth	0.1959	0.097	2.026	0.045	0.005	0.387
carheight	0.0682	0.050	1.372	0.172	-0.030	0.167
curbweight	0.2444	0.109	2.240	0.027	0.028	0.460
enginetype	0.0371	0.038	0.975	0.332	-0.038	0.113
enginesize	0.4359	0.114	3.812	0.000	0.210	0.662
boreratio	-0.0243	0.049	-0.497	0.620	-0.121	0.072
stroke	-0.1054	0.049	-2.142	0.034	-0.203	-0.008
compressionratio	0.0455	0.037	1.230	0.221	-0.028	0.119
horsepower	0.2440	0.118	2.069	0.041	0.011	0.477
peakrpm	0.0513	0.050	1.024	0.308	-0.048	0.150

Omnibus:	39.127	**Durbin-Watson:**	1.825
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	162.954
Skew:	0.886	**Prob(JB):**	4.12e-36
Kurtosis:	7.920	**Cond. No.**	49.3

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

In [35]:

```
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)
vif
```

Out[35]:

	Features	VIF
8	curbweight	58.78
6	carwidth	52.85
5	wheelbase	38.52
14	horsepower	24.44
10	enginesize	24.11
2	carbody	22.15
7	carheight	16.35
11	boreratio	15.18
12	stroke	14.06
3	drivewheel	12.37
15	peakrpm	10.32
9	enginetype	8.92
0	CarName	5.49
1	doornumber	3.49
13	compressionratio	2.96
4	enginelocation	1.35

Model 10 - Dropping the boreratio predictor as it has the highest P value (insignificane) and VIF

In [36]:

```
X = X.drop('boreratio',1)
X_train_lm = sm.add_constant(X)
lr_10 = sm.OLS(y_train, X_train_lm).fit()
lr_10.summary()
```

Out[36]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.884
Model:	OLS	**Adj. R-squared:**	0.870
Method:	Least Squares	**F-statistic:**	64.23
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	9.29e-52
Time:	10:34:07	**Log-Likelihood:**	170.68
No. Observations:	143	**AIC:**	-309.4
Df Residuals:	127	**BIC:**	-262.0
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1002	0.060	-1.678	0.096	-0.218	0.018
CarName	-0.1282	0.026	-4.980	0.000	-0.179	-0.077
doornumber	-0.0080	0.020	-0.403	0.688	-0.047	0.031
carbody	-0.0725	0.048	-1.496	0.137	-0.169	0.023
drivewheel	0.0601	0.033	1.794	0.075	-0.006	0.126
enginelocation	0.3150	0.090	3.512	0.001	0.137	0.492
wheelbase	0.0495	0.090	0.549	0.584	-0.129	0.228
carwidth	0.1862	0.094	1.972	0.051	-0.001	0.373
carheight	0.0699	0.049	1.413	0.160	-0.028	0.168
curbweight	0.2357	0.107	2.195	0.030	0.023	0.448
enginetype	0.0372	0.038	0.979	0.330	-0.038	0.112
enginesize	0.4385	0.114	3.850	0.000	0.213	0.664
stroke	-0.0983	0.047	-2.094	0.038	-0.191	-0.005
compressionratio	0.0471	0.037	1.283	0.202	-0.026	0.120
horsepower	0.2424	0.118	2.062	0.041	0.010	0.475
peakrpm	0.0568	0.049	1.168	0.245	-0.039	0.153

Omnibus:	37.208	**Durbin-Watson:**	1.820
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	146.946
Skew:	0.853	**Prob(JB):**	1.23e-32
Kurtosis:	7.664	**Cond. No.**	47.7

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

In [37]:

```
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)
vif
```

Out[37]:

	Features	VIF
8	curbweight	57.32
6	carwidth	49.41
5	wheelbase	37.95
13	horsepower	24.42
10	enginesize	24.11
2	carbody	21.44
7	carheight	16.35
11	stroke	13.49
3	drivewheel	11.69
14	peakrpm	10.09
9	enginetype	8.87
0	CarName	4.58
1	doornumber	3.43
12	compressionratio	2.93
4	enginelocation	1.33

Model 11 - Dropping the carwidth predictor as it has the highest VIF value

In [38]:

```
X = X.drop('carwidth',1)
X_train_lm = sm.add_constant(X)
lr_11 = sm.OLS(y_train, X_train_lm).fit()
lr_11.summary()
```

Out[38]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.880
Model:	OLS	**Adj. R-squared:**	0.867
Method:	Least Squares	**F-statistic:**	67.02
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	7.36e-52
Time:	10:34:08	**Log-Likelihood:**	168.52
No. Observations:	143	**AIC:**	-307.0
Df Residuals:	128	**BIC:**	-262.6
Df Model:	14		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0742	0.059	-1.261	0.210	-0.191	0.042
CarName	-0.1305	0.026	-5.018	0.000	-0.182	-0.079
doornumber	-0.0128	0.020	-0.647	0.519	-0.052	0.026
carbody	-0.0980	0.047	-2.074	0.040	-0.192	-0.005
drivewheel	0.0441	0.033	1.342	0.182	-0.021	0.109
enginelocation	0.2856	0.089	3.194	0.002	0.109	0.463
wheelbase	0.1456	0.077	1.897	0.060	-0.006	0.298
carheight	0.0652	0.050	1.304	0.195	-0.034	0.164
curbweight	0.2720	0.107	2.543	0.012	0.060	0.484
enginetype	0.0587	0.037	1.597	0.113	-0.014	0.132
enginesize	0.4367	0.115	3.792	0.000	0.209	0.664
stroke	-0.0941	0.047	-1.984	0.049	-0.188	-0.000
compressionratio	0.0662	0.036	1.847	0.067	-0.005	0.137
horsepower	0.3232	0.111	2.901	0.004	0.103	0.544
peakrpm	0.0677	0.049	1.384	0.169	-0.029	0.164

Omnibus:	29.586	**Durbin-Watson:**	1.847
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	110.726
Skew:	0.652	**Prob(JB):**	9.04e-25
Kurtosis:	7.109	**Cond. No.**	43.3

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

In [39]:

```
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)
vif
```

Out[39]:

	Features	VIF
7	curbweight	55.51
5	wheelbase	27.95
9	enginesize	24.07
12	horsepower	22.00
2	carbody	20.65
6	carheight	16.35
10	stroke	13.17
3	drivewheel	11.30
13	peakrpm	9.71
8	enginetype	7.57
0	CarName	4.57
1	doornumber	3.43
11	compressionratio	2.73
4	enginelocation	1.29

Model 12 - Dropping the wheelbase predictor as it has the highest VIF value

In [40]:

```
X = X.drop('wheelbase',1)
X_train_lm = sm.add_constant(X)
lr_12 = sm.OLS(y_train, X_train_lm).fit()
lr_12.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.877
Model:	OLS	**Adj. R-squared:**	0.864
Method:	Least Squares	**F-statistic:**	70.48
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	4.94e-52
Time:	10:34:08	**Log-Likelihood:**	166.54
No. Observations:	143	**AIC:**	-305.1
Df Residuals:	129	**BIC:**	-263.6
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0924	0.059	-1.574	0.118	-0.208	0.024
CarName	-0.1291	0.026	-4.917	0.000	-0.181	-0.077
doornumber	-0.0177	0.020	-0.890	0.375	-0.057	0.022
carbody	-0.0847	0.047	-1.794	0.075	-0.178	0.009
drivewheel	0.0650	0.031	2.079	0.040	0.003	0.127
enginelocation	0.2560	0.089	2.878	0.005	0.080	0.432
carheight	0.0964	0.048	2.024	0.045	0.002	0.191
curbweight	0.3783	0.092	4.113	0.000	0.196	0.560
enginetype	0.0541	0.037	1.458	0.147	-0.019	0.127
enginesize	0.4488	0.116	3.865	0.000	0.219	0.679
stroke	-0.0840	0.048	-1.766	0.080	-0.178	0.010
compressionratio	0.0655	0.036	1.810	0.073	-0.006	0.137
horsepower	0.2718	0.109	2.490	0.014	0.056	0.488
peakrpm	0.0770	0.049	1.568	0.119	-0.020	0.174

Omnibus:	25.502	**Durbin-Watson:**	1.848
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	97.498
Skew:	0.514	**Prob(JB):**	6.74e-22
Kurtosis:	6.912	**Cond. No.**	40.1

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

In [41]:

```
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)
vif
```

Out[41]:

	Features	VIF
6	curbweight	40.27
8	enginesize	24.06
11	horsepower	20.91
2	carbody	20.48
5	carheight	14.86
9	stroke	13.14
3	drivewheel	10.25
12	peakrpm	9.69
7	enginetype	7.41
0	CarName	4.56
1	doornumber	3.28
10	compressionratio	2.73
4	enginelocation	1.25

Model 13 - Dropping the carheight predictor as it has the highest VIF value

In [42]:

```
X = X.drop('carheight',1)
X_train_lm = sm.add_constant(X)
lr_13 = sm.OLS(y_train, X_train_lm).fit()
lr_13.summary()
```

Out[42]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.873
Model:	OLS	**Adj. R-squared:**	0.861
Method:	Least Squares	**F-statistic:**	74.25
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	4.10e-52
Time:	10:34:08	**Log-Likelihood:**	164.31
No. Observations:	143	**AIC:**	-302.6
Df Residuals:	130	**BIC:**	-264.1
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0654	0.058	-1.132	0.260	-0.180	0.049
CarName	-0.1226	0.026	-4.647	0.000	-0.175	-0.070
doornumber	-0.0188	0.020	-0.939	0.350	-0.059	0.021
carbody	-0.0483	0.044	-1.093	0.276	-0.136	0.039
drivewheel	0.0585	0.031	1.860	0.065	-0.004	0.121
enginelocation	0.2907	0.088	3.292	0.001	0.116	0.465
curbweight	0.4590	0.084	5.472	0.000	0.293	0.625
enginetype	0.0413	0.037	1.116	0.266	-0.032	0.114
enginesize	0.4227	0.117	3.620	0.000	0.192	0.654
stroke	-0.0887	0.048	-1.844	0.068	-0.184	0.006
compressionratio	0.0641	0.037	1.752	0.082	-0.008	0.137
horsepower	0.2008	0.105	1.920	0.057	-0.006	0.408
peakrpm	0.0712	0.050	1.435	0.154	-0.027	0.169

Omnibus:	28.781	**Durbin-Watson:**	1.808
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	101.345
Skew:	0.652	**Prob(JB):**	9.85e-23
Kurtosis:	6.912	**Cond. No.**	38.5

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

In [43]:

```
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)
vif
```

Out[43]:

	Features	VIF
5	curbweight	32.65
7	enginesize	23.94
10	horsepower	18.16
2	carbody	14.68
8	stroke	13.11
3	drivewheel	10.23
11	peakrpm	9.69
6	enginetype	7.35
0	CarName	4.31
1	doornumber	3.26
9	compressionratio	2.73
4	enginelocation	1.20

Model 14 - Dropping the enginetype predictor as it has the high P Value

In [44]:

```
X = X.drop('enginetype',1)
X_train_lm = sm.add_constant(X)
lr_14 = sm.OLS(y_train, X_train_lm).fit()
lr_14.summary()
```

Out[44]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.871
Model:	OLS	**Adj. R-squared:**	0.861
Method:	Least Squares	**F-statistic:**	80.73
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	8.22e-53
Time:	10:34:08	**Log-Likelihood:**	163.62
No. Observations:	143	**AIC:**	-303.2
Df Residuals:	131	**BIC:**	-267.7
Df Model:	11		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0402	0.053	-0.754	0.452	-0.146	0.065
CarName	-0.1248	0.026	-4.741	0.000	-0.177	-0.073
doornumber	-0.0165	0.020	-0.826	0.410	-0.056	0.023
carbody	-0.0443	0.044	-1.005	0.317	-0.131	0.043
drivewheel	0.0542	0.031	1.734	0.085	-0.008	0.116
enginelocation	0.2978	0.088	3.378	0.001	0.123	0.472
curbweight	0.4602	0.084	5.481	0.000	0.294	0.626
enginesize	0.4231	0.117	3.620	0.000	0.192	0.654
stroke	-0.0993	0.047	-2.103	0.037	-0.193	-0.006
compressionratio	0.0654	0.037	1.786	0.076	-0.007	0.138
horsepower	0.1995	0.105	1.906	0.059	-0.008	0.407
peakrpm	0.0731	0.050	1.472	0.143	-0.025	0.171

Omnibus:	28.581	**Durbin-Watson:**	1.820
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	90.054
Skew:	0.691	**Prob(JB):**	2.79e-20
Kurtosis:	6.634	**Cond. No.**	37.1

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

In [45]:

```
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)
vif
```

Out[45]:

	Features	VIF
5	curbweight	32.60
6	enginesize	23.79
9	horsepower	17.95
7	stroke	13.09
2	carbody	12.50
3	drivewheel	10.22
10	peakrpm	9.37
0	CarName	4.25
1	doornumber	2.94
8	compressionratio	2.73
4	enginelocation	1.20

Model 15 - Dropping the curbweight predictor as it has the high VIF Value

In [46]:

```
X = X.drop('curbweight',1)
X_train_lm = sm.add_constant(X)
lr_15 = sm.OLS(y_train, X_train_lm).fit()
lr_15.summary()
```

Out[46]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.842
Model:	OLS	**Adj. R-squared:**	0.830
Method:	Least Squares	**F-statistic:**	70.33
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	6.10e-48
Time:	10:34:08	**Log-Likelihood:**	148.86
No. Observations:	143	**AIC:**	-275.7
Df Residuals:	132	**BIC:**	-243.1
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0191	0.059	-0.326	0.745	-0.135	0.097
CarName	-0.1180	0.029	-4.064	0.000	-0.175	-0.061
doornumber	-0.0473	0.021	-2.234	0.027	-0.089	-0.005
carbody	-0.0074	0.048	-0.154	0.878	-0.103	0.088
drivewheel	0.0986	0.033	2.959	0.004	0.033	0.165
enginelocation	0.1909	0.095	2.010	0.046	0.003	0.379
enginesize	0.6842	0.118	5.804	0.000	0.451	0.917
stroke	-0.0836	0.052	-1.606	0.111	-0.187	0.019
compressionratio	0.1165	0.039	2.977	0.003	0.039	0.194
horsepower	0.4464	0.104	4.278	0.000	0.240	0.653
peakrpm	0.0402	0.054	0.739	0.461	-0.067	0.148

Omnibus:	18.335	**Durbin-Watson:**	1.848
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	37.844
Skew:	0.538	**Prob(JB):**	6.06e-09
Kurtosis:	5.279	**Cond. No.**	36.0

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

In [47]:

```
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)
vif
```

Out[47]:

	Features	VIF
5	enginesize	19.38
8	horsepower	14.68
6	stroke	12.97
2	carbody	11.67
3	drivewheel	9.40
9	peakrpm	9.27
0	CarName	4.22
1	doornumber	2.70
7	compressionratio	2.55
4	enginelocation	1.14

Model 16 - Dropping the carbody predictor as it has the high P Value

In [48]:

```
X = X.drop('carbody',1)
X_train_lm = sm.add_constant(X)
lr_16 = sm.OLS(y_train, X_train_lm).fit()
lr_16.summary()
```

Out[48]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.842
Model:	OLS	**Adj. R-squared:**	0.831
Method:	Least Squares	**F-statistic:**	78.72
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	6.74e-49
Time:	10:34:08	**Log-Likelihood:**	148.85
No. Observations:	143	**AIC:**	-277.7
Df Residuals:	133	**BIC:**	-248.1
Df Model:	9		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0250	0.044	-0.566	0.572	-0.113	0.062
CarName	-0.1180	0.029	-4.080	0.000	-0.175	-0.061
doornumber	-0.0453	0.017	-2.732	0.007	-0.078	-0.012
drivewheel	0.0992	0.033	3.009	0.003	0.034	0.164
enginelocation	0.1932	0.093	2.067	0.041	0.008	0.378
enginesize	0.6841	0.117	5.825	0.000	0.452	0.916
stroke	-0.0832	0.052	-1.606	0.111	-0.186	0.019
compressionratio	0.1160	0.039	2.986	0.003	0.039	0.193
horsepower	0.4457	0.104	4.291	0.000	0.240	0.651
peakrpm	0.0396	0.054	0.733	0.465	-0.067	0.147

Omnibus:	18.131	**Durbin-Watson:**	1.847
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	37.161
Skew:	0.534	**Prob(JB):**	8.52e-09
Kurtosis:	5.257	**Cond. No.**	33.3

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

In [49]:

```
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)
vif
```

Out[49]:

	Features	VIF
4	enginesize	18.41
7	horsepower	14.33
5	stroke	11.78
2	drivewheel	9.37
8	peakrpm	7.58
0	CarName	3.44
6	compressionratio	2.50
1	doornumber	2.15
3	enginelocation	1.11

Model 17 - Dropping the peakrpm predictor as it has the high P Value and VIF Value

In [50]:

```
X = X.drop('peakrpm',1)
X_train_lm = sm.add_constant(X)
lr_17 = sm.OLS(y_train, X_train_lm).fit()
lr_17.summary()
```

Out[50]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.841
Model:	OLS	**Adj. R-squared:**	0.832
Method:	Least Squares	**F-statistic:**	88.80
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	9.01e-50
Time:	10:34:08	**Log-Likelihood:**	148.56
No. Observations:	143	**AIC:**	-279.1
Df Residuals:	134	**BIC:**	-252.5
Df Model:	8		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0078	0.037	-0.209	0.835	-0.082	0.066
CarName	-0.1218	0.028	-4.288	0.000	-0.178	-0.066
doornumber	-0.0441	0.016	-2.679	0.008	-0.077	-0.012
drivewheel	0.1010	0.033	3.077	0.003	0.036	0.166
enginelocation	0.2028	0.092	2.196	0.030	0.020	0.385
enginesize	0.6361	0.097	6.539	0.000	0.444	0.828
stroke	-0.0759	0.051	-1.495	0.137	-0.176	0.024
compressionratio	0.1071	0.037	2.907	0.004	0.034	0.180
horsepower	0.4818	0.091	5.278	0.000	0.301	0.662

Omnibus:	19.584	**Durbin-Watson:**	1.843
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	39.832
Skew:	0.586	**Prob(JB):**	2.24e-09
Kurtosis:	5.305	**Cond. No.**	26.6

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

In [51]:

```
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)
vif
```

Out[51]:

	Features	VIF
4	enginesize	14.16
7	horsepower	12.00
2	drivewheel	8.84
5	stroke	7.55
0	CarName	3.31
6	compressionratio	2.28
1	doornumber	2.06
3	enginelocation	1.09

Model 18 - Dropping the enginesize predictor as it has the high P Value and VIF Value

In [52]:

```
X = X.drop('enginesize',1)
X_train_lm = sm.add_constant(X)
lr_18 = sm.OLS(y_train, X_train_lm).fit()
lr_18.summary()
```

Out[52]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.791
Model:	OLS	**Adj. R-squared:**	0.780
Method:	Least Squares	**F-statistic:**	72.84
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	9.67e-43
Time:	10:34:08	**Log-Likelihood:**	128.76
No. Observations:	143	**AIC:**	-241.5
Df Residuals:	135	**BIC:**	-217.8
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0024	0.043	0.056	0.955	-0.082	0.087
CarName	-0.1464	0.032	-4.543	0.000	-0.210	-0.083
doornumber	-0.0741	0.018	-4.097	0.000	-0.110	-0.038
drivewheel	0.1334	0.037	3.590	0.000	0.060	0.207
enginelocation	0.1833	0.106	1.735	0.085	-0.026	0.392
stroke	-0.0218	0.057	-0.380	0.705	-0.135	0.092
compressionratio	0.1700	0.041	4.176	0.000	0.089	0.250
horsepower	0.9549	0.064	14.984	0.000	0.829	1.081

Omnibus:	34.934	**Durbin-Watson:**	1.640
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	72.555
Skew:	1.054	**Prob(JB):**	1.76e-16
Kurtosis:	5.780	**Cond. No.**	19.3

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

In [53]:

```
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)
vif
```

Out[53]:

	Features	VIF
2	drivewheel	8.58
4	stroke	7.01
6	horsepower	4.47
0	CarName	3.25
5	compressionratio	2.13
1	doornumber	1.90
3	enginelocation	1.09

Model 19 - Dropping the stroke predictor as it has the high P Value and VIF Value

In [54]:

```
X = X.drop('stroke',1)
X_train_lm = sm.add_constant(X)
lr_19 = sm.OLS(y_train, X_train_lm).fit()
lr_19.summary()
```

Out[54]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.790
Model:	OLS	**Adj. R-squared:**	0.781
Method:	Least Squares	**F-statistic:**	85.50
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	1.07e-43
Time:	10:34:08	**Log-Likelihood:**	128.69
No. Observations:	143	**AIC:**	-243.4
Df Residuals:	136	**BIC:**	-222.6
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0092	0.030	-0.307	0.759	-0.068	0.050
CarName	-0.1440	0.032	-4.570	0.000	-0.206	-0.082
doornumber	-0.0743	0.018	-4.121	0.000	-0.110	-0.039
drivewheel	0.1336	0.037	3.609	0.000	0.060	0.207
enginelocation	0.1881	0.105	1.800	0.074	-0.019	0.395
compressionratio	0.1664	0.039	4.214	0.000	0.088	0.245
horsepower	0.9516	0.063	15.119	0.000	0.827	1.076

Omnibus:	34.637	**Durbin-Watson:**	1.634
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	71.741
Skew:	1.046	**Prob(JB):**	2.64e-16
Kurtosis:	5.768	**Cond. No.**	17.9

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

In [55]:

```
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)
vif
```

Out[55]:

	Features	VIF
2	drivewheel	7.68
5	horsepower	4.30
0	CarName	2.74
4	compressionratio	1.96
1	doornumber	1.76
3	enginelocation	1.05

Model 20 - Dropping the drivewheel predictor as it has the high P Value and VIF Value

In [56]:

```
X = X.drop('drivewheel',1)
X_train_lm = sm.add_constant(X)
lr_20 = sm.OLS(y_train, X_train_lm).fit()
lr_20.summary()
```

Out[56]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.770
Model:	OLS	**Adj. R-squared:**	0.762
Method:	Least Squares	**F-statistic:**	91.93
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	5.06e-42
Time:	10:34:08	**Log-Likelihood:**	122.15
No. Observations:	143	**AIC:**	-232.3
Df Residuals:	137	**BIC:**	-214.5
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0390	0.028	1.398	0.164	-0.016	0.094
CarName	-0.1487	0.033	-4.529	0.000	-0.214	-0.084
doornumber	-0.0669	0.019	-3.582	0.000	-0.104	-0.030
enginelocation	0.1807	0.109	1.658	0.100	-0.035	0.396
compressionratio	0.2180	0.038	5.675	0.000	0.142	0.294
horsepower	1.0733	0.055	19.365	0.000	0.964	1.183

Omnibus:	36.381	**Durbin-Watson:**	1.559
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	78.682
Skew:	1.079	**Prob(JB):**	8.21e-18
Kurtosis:	5.923	**Cond. No.**	15.8

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

In [57]:

```
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)
vif
```

Out[57]:

	Features	VIF
0	CarName	2.36
4	horsepower	2.32
1	doornumber	1.57
3	compressionratio	1.48
2	enginelocation	1.05

Model 21 - Dropping the enginelocation predictor as it has the high P Value

In [58]:

```
X = X.drop('enginelocation',1)
X_train_lm = sm.add_constant(X)
lr_21 = sm.OLS(y_train, X_train_lm).fit()
lr_21.summary()
```

Out[58]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.766
Model:	OLS	**Adj. R-squared:**	0.759
Method:	Least Squares	**F-statistic:**	112.8
Date:	Wed, 02 Oct 2019	**Prob (F-statistic):**	1.72e-42
Time:	10:34:08	**Log-Likelihood:**	120.73
No. Observations:	143	**AIC:**	-231.5
Df Residuals:	138	**BIC:**	-216.6
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.0316	0.028	1.141	0.256	-0.023	0.086
CarName	-0.1443	0.033	-4.381	0.000	-0.209	-0.079
doornumber	-0.0640	0.019	-3.419	0.001	-0.101	-0.027
compressionratio	0.2203	0.039	5.703	0.000	0.144	0.297
horsepower	1.0936	0.054	20.104	0.000	0.986	1.201

Omnibus:	31.024	**Durbin-Watson:**	1.583
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	61.723
Skew:	0.954	**Prob(JB):**	3.95e-14
Kurtosis:	5.592	**Cond. No.**	8.19

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

In [59]:

```
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)
vif
```

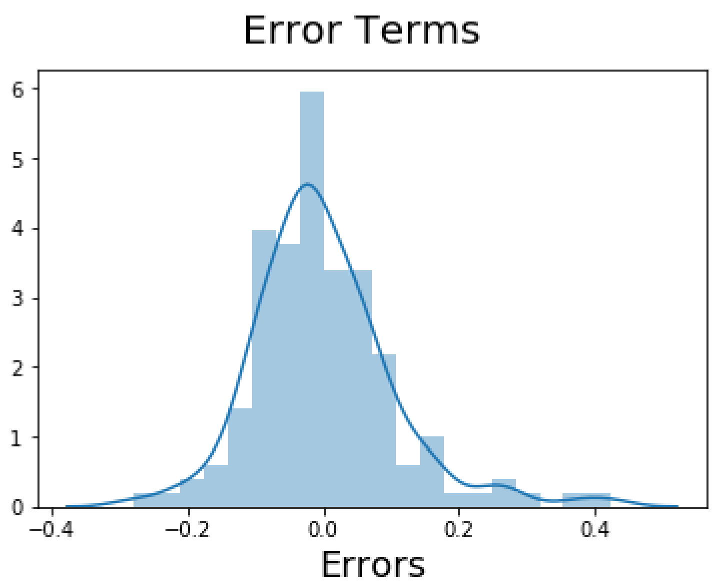
Out[59]:

	Features	VIF
0	CarName	2.35
3	horsepower	2.25
1	doornumber	1.57
2	compressionratio	1.48

Step 7 : Residual Analysis of the error data (train)

In [60]:

```
y_train_residual = lr_21.predict(X_train_lm)

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



Step 8 : Making the predictions using test data

In [61]:

```
master_df_test[num_columns_list] = scaler.fit_transform(master_df_test[num_columns_list
])
master_df_test[non_num_columns_list] = scaler.fit_transform(master_df_test[non_num_colu
mns_list])
master_df_test.describe()
```

Out[61]:

	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewh
count	62.000000	62.000000	62.000000	62.000000	62.000000	62.000000	62.000(
mean	0.583871	0.566532	0.887097	0.177419	0.435484	0.625000	0.701€
std	0.271724	0.311481	0.319058	0.385142	0.499868	0.225205	0.263:
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000(
25%	0.400000	0.375000	1.000000	0.000000	0.000000	0.500000	0.500(
50%	0.600000	0.562500	1.000000	0.000000	0.000000	0.750000	0.500(
75%	0.800000	0.833333	1.000000	0.000000	1.000000	0.750000	1.000(
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000(

8 rows × 25 columns

In [62]:

```
## Dividing between x_test and y_test

test_columns = list(master_df_test.columns)

y_test = master_df_test['price']
test_columns.remove('price')
x_test = master_df_test[test_columns]
```

In [63]:

```
y_test.head()
```

Out[63]:

```
car_ID
161    0.058474
187    0.077398
60     0.086148
166    0.097473
141    0.055099
Name: price, dtype: float64
```

In [64]:

```
x_test.head()
```

Out[64]:

	Insuranceriskfactor	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
car_ID							
161	0.4	0.833333	1.0	0.0	0.0	0.75	0.5
187	0.8	0.958333	1.0	0.0	0.0	0.75	0.5
60	0.6	0.375000	1.0	0.0	1.0	0.50	0.5
166	0.6	0.833333	1.0	0.0	1.0	0.75	1.0
141	0.8	0.791667	1.0	0.0	1.0	0.50	0.0

5 rows × 24 columns

In [65]:

```
## add constant variable to x_test
x_test_sm = sm.add_constant(x_test)

## Keeping only the relevant predictors from the final lr_20 model
x_test_pred = x_test_sm[['const', 'CarName', 'doornumber', 'compressionratio', 'horsepo
wer']]

## Making the predictions on the test set
y_test_pred = lr_21.predict(x_test_pred)
```

In [66]:

```
y_test_pred.head()
```

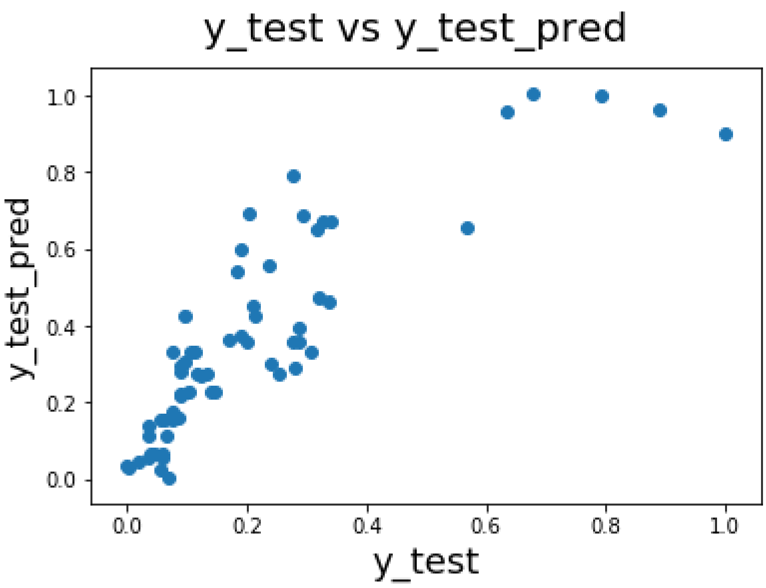
Out[66]:

```
car_ID
161    0.065901
187    0.153693
60     0.161308
166    0.303734
141    0.024957
dtype: float64
```

Step 9 : Model Evaluation

In [67]:

```
# Plotting y_test and y_pred to understand the spread

fig = plt.figure()
plt.scatter(y_test, y_test_pred)
fig.suptitle('y_test vs y_test_pred', fontsize = 20)
plt.xlabel('y_test', fontsize = 18)
plt.ylabel('y_test_pred', fontsize = 16)
plt.show()
```



Step 10 : Calculate the R-square value

In [68]:

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

Out[68]:

```
0.07040807219687462
```

In [ ]:

```

```