Multiple Linear Regression Assignment

Problem Statement

A Chinese automobile company Geely Auto aspires to enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.

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

Which variables are significant in predicting the price of a car How well those variables describe the price of a car Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the Americal market.

Business Goal

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

Step 1: lets now load data and understand the data

In [1]:
```
# Supress Warnings

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

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
GeelyAuto = pd.read_csv('CarPrice_Assignment.csv')
```

In [4]:
```
# Lets check the head of the dataset
GeelyAuto.head()
```

Out[4]:

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

5 rows × 26 columns

In [5]:
```
# Lets check the data description and info
GeelyAuto.info()
```

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

In [6]:
```
GeelyAuto.describe()
```

Out[6]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	engi
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.9
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.6
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.0
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.0
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.0
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.0
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.0

Step 2: Data Preparation

You can see that your dataset has many columns with categorical data.

But in order to fit a regression line, we would need numerical values and not string. Hence, we need to convert them to 1s and 0s.

In [7]:
```
GeelyAuto.set_index('car_ID',inplace=True)
GeelyAuto.head()
```

Out[7]:

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

5 rows × 25 columns

In [8]:
```
# Lets now split carname and consider only car company name
GeelyAuto['CarName']=GeelyAuto['CarName'].apply(lambda x: x.split(' ')[0])
```

In [9]:
```
GeelyAuto['CarName'].unique()
```

Out[9]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'maxda', 'mazda', 'buick', 'mercury',
       'mitsubishi', 'Nissan', 'nissan', 'peugeot', 'plymouth', 'porsche',
       'porcshce', 'renault', 'saab', 'subaru', 'toyota', 'toyouta',
       'vokswagen', 'volkswagen', 'vw', 'volvo'], dtype=object)
```

In [10]:
```
# We see some names from GeelyAuto CarName column having redudant or some name is
GeelyAuto['CarName']=GeelyAuto['CarName'].apply(lambda x: 'volkswagen' if x=='vok
GeelyAuto['CarName']=GeelyAuto['CarName'].apply(lambda x: 'mazda' if x=='maxda' e
GeelyAuto['CarName']=GeelyAuto['CarName'].apply(lambda x: 'nissan' if x=='Nissan'
GeelyAuto['CarName']=GeelyAuto['CarName'].apply(lambda x: 'porsche' if x=='porcsh
GeelyAuto['CarName']=GeelyAuto['CarName'].apply(lambda x: 'toyota' if x=='toyouta
GeelyAuto['CarName']=GeelyAuto['CarName'].apply(lambda x: 'volkswagen' if x=='vw'
```

In [11]:
```
GeelyAuto['CarName'].unique()
```

Out[11]:
```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'buick', 'mercury', 'mitsubishi',
       'nissan', 'peugeot', 'plymouth', 'porsche', 'renault', 'saab',
       'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

In [12]:
```
#Now lets check for other columns
GeelyAuto.nunique()
```

Out[12]:
```
symboling             6
CarName              22
fueltype              2
aspiration            2
doornumber            2
carbody               5
drivewheel            3
enginelocation        2
wheelbase            53
carlength            75
carwidth             44
carheight            49
curbweight          171
enginetype            7
cylindernumber        7
enginesize           44
fuelsystem            8
boreratio            38
stroke               37
compressionratio     32
horsepower           59
peakrpm              23
citympg              29
highwaympg           30
price               189
dtype: int64
```

In [13]:
```
#Now to change categorical data for this lets check unique data
GeelyAuto.nunique()
```

Out[13]:
```
symboling             6
CarName              22
fueltype              2
aspiration            2
doornumber            2
carbody               5
drivewheel            3
enginelocation        2
wheelbase            53
carlength            75
carwidth             44
carheight            49
curbweight          171
enginetype            7
cylindernumber        7
enginesize           44
fuelsystem            8
boreratio            38
stroke               37
compressionratio     32
horsepower           59
peakrpm              23
citympg              29
highwaympg           30
price               189
dtype: int64
```

In [14]:
```
#Now lets set all categorycall data with 2 possiblities with 1 or 0
#array(['gas', 'diesel'], dtype=object) for fueltype
def decodeFuelType(fuelType):
    if fuelType=='gas':
        return 1
    else:
        return 0
GeelyAuto['fueltype']=GeelyAuto['fueltype'].apply(decodeFuelType)
```

In [15]:
```
GeelyAuto['enginetype'].unique()
```

Out[15]:
```
array(['dohc', 'ohcv', 'ohc', 'l', 'rotor', 'ohcf', 'dohcv'], dtype=object)
```

In [16]:
```
# array(['std', 'turbo'], dtype=object) for aspiration
def decodeType(x):
    if x=='std':
        return 1
    else:
        return 0
GeelyAuto['aspiration']=GeelyAuto['aspiration'].apply(decodeType)

# array(['two', 'four'], dtype=object) for doornumber
def decodeType(x):
    if x=='two':
        return 1
    else:
        return 0
GeelyAuto['doornumber']=GeelyAuto['doornumber'].apply(decodeType)
# array(['front', 'rear'], dtype=object) for enginelocation
def decodeType(x):
    if x=='front':
        return 1
    else:
        return 0
GeelyAuto['enginelocation']=GeelyAuto['enginelocation'].apply(decodeType)
```

Dummy Variables

In [17]:
```
#now to create some dummy variables for CarName category
CarName=pd.get_dummies(GeelyAuto['CarName'],drop_first = True,prefix='CarName',pr
#CarName.drop('Nissan',inplace=True,axis=1)
CarName.head()
```

Out[17]:

car_ID	CarName_audi	CarName_bmw	CarName_buick	CarName_chevrolet	CarName_dodge	CarNa
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	1	0	0	0	0	
5	1	0	0	0	0	

5 rows × 21 columns

In [18]:
```
GeelyAuto = pd.concat([GeelyAuto,CarName],axis=1)
```

In [19]:
```
#now to create some dummy variables for carbody category
carbody=pd.get_dummies(GeelyAuto['carbody'],drop_first = True,prefix='carbody',pr
#carbody.head()
GeelyAuto = pd.concat([GeelyAuto,carbody],axis=1)

#now to create some dummy variables for drivewheel category
drivewheel=pd.get_dummies(GeelyAuto['drivewheel'],drop_first = True,prefix='drive
#drivewheel.head()
GeelyAuto = pd.concat([GeelyAuto,drivewheel],axis=1)
#now to create some dummy variables for enginetype category
enginetype=pd.get_dummies(GeelyAuto['enginetype'],drop_first = True,prefix='engin
#enginetype.head()
GeelyAuto = pd.concat([GeelyAuto,enginetype],axis=1)

#now to create some dummy variables for cylindernumber category
cylindernumber=pd.get_dummies(GeelyAuto['cylindernumber'],drop_first = True,prefi
#cylindernumber.head()
GeelyAuto = pd.concat([GeelyAuto,cylindernumber],axis=1)

#now to create some dummy variables for fuelsystem category
fuelsystem=pd.get_dummies(GeelyAuto['fuelsystem'],drop_first = True,prefix='fuels
#fuelsystem.head()
GeelyAuto = pd.concat([GeelyAuto,fuelsystem],axis=1)
```

In [20]:
```
GeelyAuto.describe()
```

Out[20]:

	symboling	fueltype	aspiration	doornumber	enginelocation	wheelbase	carlength
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	0.902439	0.819512	0.439024	0.985366	98.756585	174.049268
std	1.245307	0.297446	0.385535	0.497483	0.120377	6.021776	12.337289
min	-2.000000	0.000000	0.000000	0.000000	0.000000	86.600000	141.100000
25%	0.000000	1.000000	1.000000	0.000000	1.000000	94.500000	166.300000
50%	1.000000	1.000000	1.000000	0.000000	1.000000	97.000000	173.200000
75%	2.000000	1.000000	1.000000	1.000000	1.000000	102.400000	183.100000
max	3.000000	1.000000	1.000000	1.000000	1.000000	120.900000	208.100000

8 rows × 65 columns

In [21]:
```
GeelyAuto.drop(['CarName','carbody','drivewheel','enginetype','cylindernumber','f
```

In [22]:
```
GeelyAuto.head()
```

Out[22]:

	symboling	fueltype	aspiration	doornumber	enginelocation	wheelbase	carlength	carwidt
car_ID								
1	3	1	1	1	1	88.6	168.8	64.
2	3	1	1	1	1	88.6	168.8	64.
3	1	1	1	1	1	94.5	171.2	65.
4	2	1	1	0	1	99.8	176.6	66.
5	2	1	1	0	1	99.4	176.6	66.

5 rows × 65 columns

In [23]:
```
GeelyAuto.nunique()
```

Out[23]:
```
symboling                  6
fueltype                   2
aspiration                 2
doornumber                 2
enginelocation             2
wheelbase                 53
carlength                 75
carwidth                  44
carheight                 49
curbweight               171
enginesize                44
boreratio                 38
stroke                    37
compressionratio          32
horsepower                59
peakrpm                   23
citympg                   29
highwaympg                30
price                    189
CarName_audi               2
CarName_bmw                2
CarName_buick              2
CarName_chevrolet          2
CarName_dodge              2
CarName_honda              2
CarName_isuzu              2
CarName_jaguar             2
CarName_mazda              2
CarName_mercury            2
CarName_mitsubishi         2
                        ... 
CarName_saab               2
CarName_subaru             2
CarName_toyota             2
CarName_volkswagen         2
CarName_volvo              2
carbody_hardtop            2
carbody_hatchback          2
carbody_sedan              2
carbody_wagon              2
drivewheel_fwd             2
drivewheel_rwd             2
enginetype_dohcv           2
enginetype_l               2
enginetype_ohc             2
enginetype_ohcf            2
enginetype_ohcv            2
enginetype_rotor           2
cylindernumber_five        2
cylindernumber_four        2
cylindernumber_six         2
cylindernumber_three       2
cylindernumber_twelve      2
cylindernumber_two         2
```

```
fuelsystem_2bbl          2
fuelsystem_4bbl          2
fuelsystem_idi           2
fuelsystem_mfi           2
fuelsystem_mpfi          2
fuelsystem_spdi          2
fuelsystem_spfi          2
Length: 65, dtype: int64
```

Step 3: Splitting the Data into Training and Testing Sets

As you know, the first basic step for regression is performing a train-test split.

In [24]:
```
from sklearn.model_selection import train_test_split

# We specify this so that the train and test data set always have the same rows, r
np.random.seed(0)
df_train, df_test = train_test_split(GeelyAuto, train_size = 0.7, test_size = 0.3
```

Rescaling the Features

As you saw in the in our leactures of Simple Linear Regression, scaling doesn't impact our model. Here we can see that except for curbweight and peek rpm, all the columns have small integer values or categorical data. So it is extremely important to rescale the variables so that they have a comparable scale. If we don't have comparable scales, then some of the coefficients as obtained by fitting the regression model might be very large or very small as compared to the other coefficients. This might become very annoying at the time of model evaluation. So it is advised to use standardization or normalization so that the units of the coefficients obtained are all on the same scale. As you know, there are two common ways of rescaling:

1. Min-Max scaling
2. Standardisation (mean-0, sigma-1)

Option chosen will use MinMax scaling.

In [25]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [26]:
```
scaler = MinMaxScaler()
```

In [27]:
```
num_vars = ['symboling','wheelbase','carlength','carwidth','carheight','curbweigh
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

In [28]:
```
df_train.head()
```

Out[28]:

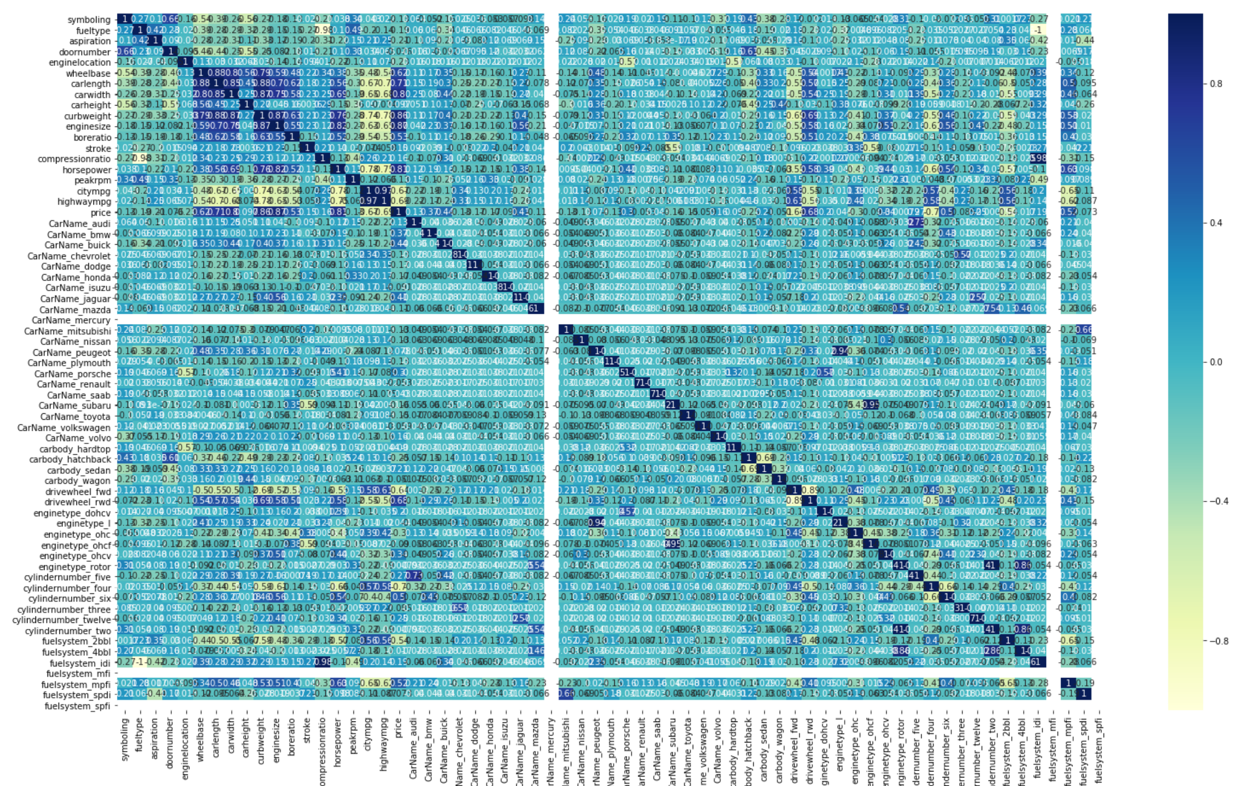
car_ID	symboling	fueltype	aspiration	doornumber	enginelocation	wheelbase	carlength	carwidt
123	0.6	1	1	0	1	0.244828	0.426016	0.29166
126	1.0	1	1	1	1	0.272414	0.452033	0.66666
167	0.6	1	1	1	1	0.272414	0.448780	0.30833
2	1.0	1	1	1	1	0.068966	0.450407	0.31666
200	0.2	1	0	0	1	0.610345	0.775610	0.57500

5 rows × 65 columns

In [ ]:

In [29]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [30]:
```
# Let's check the correlation coefficients to see which variables are highly corre

plt.figure(figsize = (26, 15))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



In [31]:
```
# With so many variables in our data set its very difficult to infer anything.Let
col = ['symboling','wheelbase','carlength','carwidth','carheight','curbweight','e
plt.figure(figsize = (26, 15))
sns.heatmap(df_train[col].corr(), annot = True, cmap="YlGnBu")
plt.show()
```

In [32]:
```
df_train.corr()['price']
```

Out[32]:
```
symboling              -0.129859
fueltype               -0.191150
aspiration             -0.206540
doornumber             -0.075936
enginelocation         -0.226217
wheelbase               0.622591
carlength               0.713749
carwidth                0.799380
carheight               0.096631
curbweight              0.861860
enginesize              0.867915
boreratio               0.533591
stroke                  0.152820
compressionratio        0.160847
horsepower              0.806183
peakrpm                -0.127431
citympg                -0.674290
highwaympg             -0.688389
price                   1.000000
CarName_audi            0.131449
CarName_bmw             0.371790
CarName_buick           0.437268
CarName_chevrolet      -0.132643
CarName_dodge          -0.165673
CarName_honda          -0.172485
CarName_isuzu          -0.091266
CarName_jaguar          0.405372
CarName_mazda          -0.105680
CarName_mercury              NaN
CarName_mitsubishi     -0.134087
                          ...
CarName_saab            0.042719
CarName_subaru         -0.164373
CarName_toyota         -0.148968
CarName_volkswagen     -0.059238
CarName_volvo           0.161924
carbody_hardtop         0.089735
carbody_hatchback      -0.252484
carbody_sedan           0.205018
carbody_wagon          -0.051173
drivewheel_fwd         -0.635202
drivewheel_rwd          0.677169
enginetype_dohcv        0.197875
enginetype_l            0.044246
enginetype_ohc         -0.297108
enginetype_ohcf        -0.089985
enginetype_ohcv         0.339468
enginetype_rotor       -0.000793
cylindernumber_five     0.271430
cylindernumber_four    -0.695256
cylindernumber_six      0.500613
cylindernumber_three   -0.085274
cylindernumber_twelve   0.247489
cylindernumber_two     -0.000793
```

```
fuelsystem_2bbl        -0.537919
fuelsystem_4bbl        -0.017148
fuelsystem_idi          0.191150
fuelsystem_mfi               NaN
fuelsystem_mpfi         0.519993
fuelsystem_spdi        -0.073240
fuelsystem_spfi              NaN
Name: price, Length: 65, dtype: float64
```

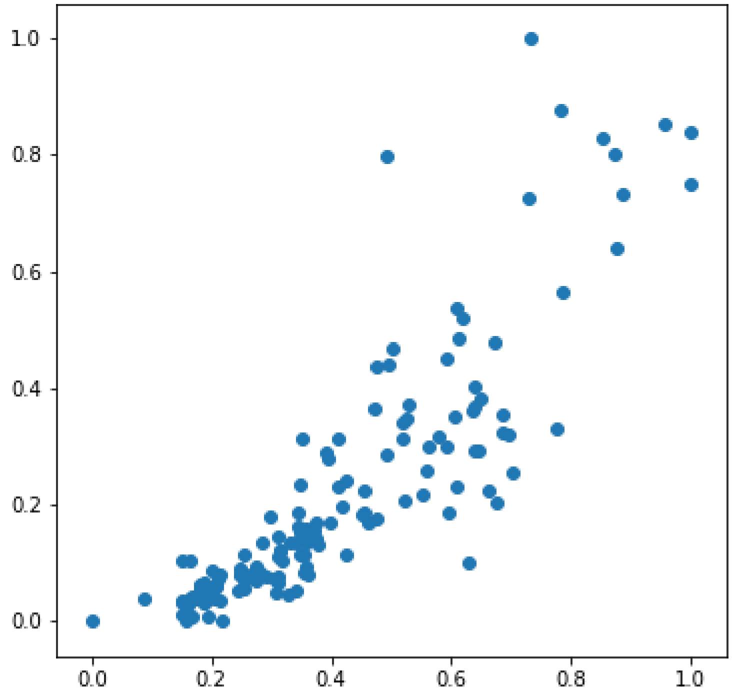
In [33]:

```
#We see some relation between columns and Price so to narrow down the correlation
mask = df_train.corr()['price']>=0.3
df_train.corr()[mask]['price']
```

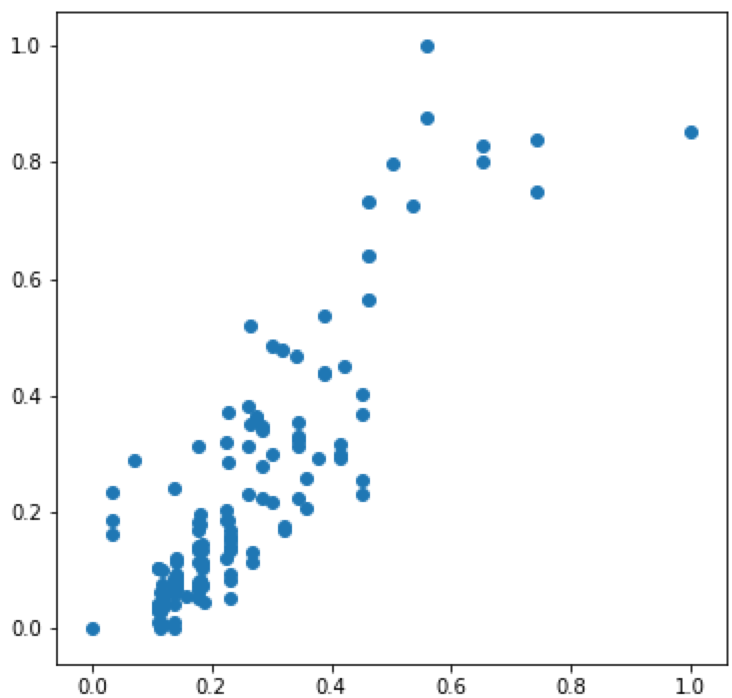
Out[33]:

```
wheelbase             0.622591
carlength             0.713749
carwidth              0.799380
curbweight            0.861860
enginesize            0.867915
boreratio             0.533591
horsepower            0.806183
price                 1.000000
CarName_bmw           0.371790
CarName_buick         0.437268
CarName_jaguar        0.405372
CarName_porsche       0.302801
drivewheel_rwd        0.677169
enginetype_ohcv       0.339468
cylindernumber_six    0.500613
fuelsystem_mpfi       0.519993
Name: price, dtype: float64
```

In [34]:
```
plt.figure(figsize=[6,6])
plt.scatter(df_train.curbweight, df_train.price)
plt.show()
```



In [35]:
```
plt.figure(figsize=[6,6])
plt.scatter(df_train.enginesize, df_train.price)
plt.show()
```



if we see the top few scatter plot we see some linear

relation with the price now lets go further and analyse some data.

Dividing into X and Y sets for the model building

In [36]:
```
y_train = df_train.pop('price')
X_train = df_train
```

Step 4: Building a linear model

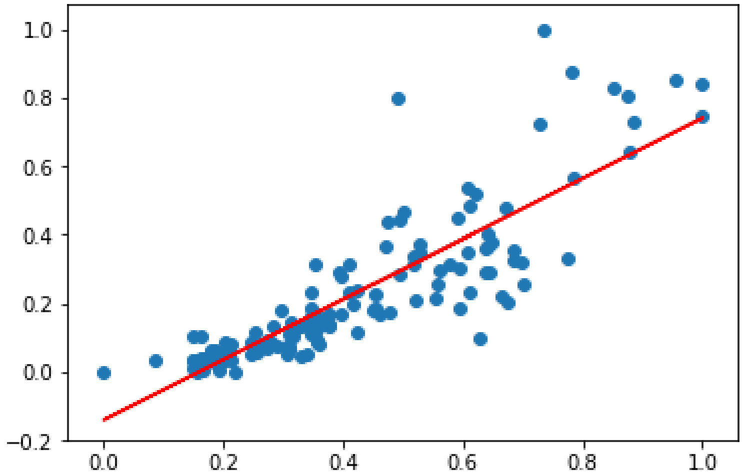
Fit a regression line through the training data using statsmodels. Remember that in statsmodels, you need to explicitly fit a constant using sm.add_constant(X) because if we don't perform this step, statsmodels fits a regression line passing through the origin, by default.

In [37]:
```
import statsmodels.api as sm

# Add a constant
X_train_lm = sm.add_constant(X_train[['curbweight']])

# Create a first fitted model with curbweight and price
lr = sm.OLS(y_train, X_train_lm).fit()
```

In [38]:
```
# Now lets check the parameters obtained

lr.params
```

Out[38]:
```
const        -0.139568
curbweight    0.879863
dtype: float64
```

In [39]:
```
# Let's visualise the data with a scatter plot and the fitted regression line
plt.scatter(X_train_lm.iloc[:, 1], y_train)
plt.plot(X_train_lm.iloc[:, 1], -0.139568 + 0.879863*X_train_lm.iloc[:, 1], 'r')
plt.show()
```



In [40]:
```
# Print a summary of the linear regression model obtained considering curbweight
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.743
Model:                            OLS   Adj. R-squared:                  0.741
Method:                 Least Squares   F-statistic:                     407.2
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           2.06e-43
Time:                        07:35:48   Log-Likelihood:                 114.04
No. Observations:                 143   AIC:                            -224.1
Df Residuals:                     141   BIC:                            -218.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1396      0.020     -6.974      0.000      -0.179      -0.100
curbweight     0.8799      0.044     20.180      0.000       0.794       0.966
==============================================================================
Omnibus:                       51.679   Durbin-Watson:                   1.690
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              220.291
Skew:                           1.232   Prob(JB):                     1.46e-48
Kurtosis:                       8.559   Cond. No.                         5.57
==============================================================================

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

Add another variable

In [41]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train[['curbweight','enginesize']])

# Create a first fitted model with curbweight and price
lr = sm.OLS(y_train, X_train_lm).fit()
```

In [42]:
```
lr.params
```

Out[42]:
```
const        -0.128113
curbweight    0.450496
enginesize    0.678162
dtype: float64
```

In [43]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.802
Model:                            OLS   Adj. R-squared:                  0.799
Method:                 Least Squares   F-statistic:                     284.0
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           5.31e-50
Time:                        07:35:48   Log-Likelihood:                 132.84
No. Observations:                 143   AIC:                            -259.7
Df Residuals:                     140   BIC:                            -250.8
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1281      0.018     -7.239      0.000      -0.163      -0.093
curbweight     0.4505      0.076      5.890      0.000       0.299       0.602
enginesize     0.6782      0.105      6.489      0.000       0.472       0.885
==============================================================================
Omnibus:                       36.002   Durbin-Watson:                   1.830
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               85.998
Skew:                           1.023   Prob(JB):                     2.12e-19
Kurtosis:                       6.202   Cond. No.                         17.3
==============================================================================

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

We see some increase on the adjusted R square so this is a value addition.

Let move forward to use RFE for feature selection and see if we can create a better model

In [44]:
```
#Build a linear model considering all the variables to analyse the impact

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.params
```

Out[44]:
```
const                 2.306669e-02
symboling            -9.879522e-03
fueltype             -1.260443e-01
aspiration           -8.113676e-02
doornumber           -1.271982e-02
enginelocation       -1.903334e-01
wheelbase             2.367996e-01
carlength            -1.752292e-01
carwidth              2.597715e-01
carheight            -1.762854e-01
curbweight            3.291759e-01
enginesize            2.010441e+00
boreratio            -6.214539e-01
stroke               -1.812934e-01
compressionratio     -3.611938e-01
horsepower           -2.005293e-01
peakrpm               1.878108e-01
citympg              -5.156732e-02
highwaympg            1.273365e-01
CarName_audi          6.967675e-02
CarName_bmw           3.396067e-01
CarName_buick         4.930883e-02
CarName_chevrolet    -5.223870e-02
CarName_dodge        -1.089892e-01
CarName_honda        -6.885945e-02
CarName_isuzu         2.295335e-03
CarName_jaguar       -1.331677e-01
CarName_mazda         2.447154e-02
CarName_mercury      -5.682073e-15
CarName_mitsubishi   -1.273068e-01
                          ...
CarName_saab          2.259475e-01
CarName_subaru       -4.025895e-02
CarName_toyota        1.648532e-02
CarName_volkswagen    2.285186e-02
CarName_volvo         1.069129e-01
carbody_hardtop      -9.714369e-02
carbody_hatchback    -1.058254e-01
carbody_sedan        -8.663686e-02
carbody_wagon        -6.762364e-02
drivewheel_fwd       -3.432830e-03
drivewheel_rwd        2.624883e-02
enginetype_dohcv      2.378634e-01
enginetype_l          2.070738e-01
enginetype_ohc        1.040682e-03
enginetype_ohcf       1.731411e-01
enginetype_ohcv      -2.752914e-02
```

```
enginetype_rotor         3.960833e-01
cylindernumber_five      2.665612e-01
cylindernumber_four      4.385606e-01
cylindernumber_six       1.188905e-01
cylindernumber_three     4.954953e-01
cylindernumber_twelve   -3.562871e-01
cylindernumber_two       3.960833e-01
fuelsystem_2bbl          2.322706e-02
fuelsystem_4bbl         -4.880078e-02
fuelsystem_idi           1.491109e-01
fuelsystem_mfi           0.000000e+00
fuelsystem_mpfi         -1.610708e-02
fuelsystem_spdi         -2.618883e-02
fuelsystem_spfi          0.000000e+00
Length: 65, dtype: float64
```

In [45]:
```
print(lr_1.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  price   R-squared:                       0.975
Model:                            OLS   Adj. R-squared:                  0.958
Method:                 Least Squares   F-statistic:                     57.59
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           1.40e-49
Time:                        07:35:48   Log-Likelihood:                 280.02
No. Observations:                 143   AIC:                            -444.0
Df Residuals:                      85   BIC:                            -272.2
Df Model:                          57                                         
Covariance Type:            nonrobust                                         
================================================================================
==========
                     coef    std err          t      P>|t|      [0.025
    0.975]
--------------------------------------------------------------------------------
----------
const              0.0231      0.142      0.162      0.872      -0.260
     0.306
symboling         -0.0099      0.037     -0.265      0.792      -0.084
     0.064
fueltype          -0.1260      0.100     -1.265      0.209      -0.324
     0.072
aspiration        -0.0811      0.029     -2.778      0.007      -0.139
    -0.023
doornumber        -0.0127      0.015     -0.841      0.403      -0.043
     0.017
enginelocation    -0.1903      0.058     -3.285      0.001      -0.306
    -0.075
wheelbase          0.2368      0.093      2.542      0.013       0.052
     0.422
carlength         -0.1752      0.100     -1.747      0.084      -0.375
     0.024
carwidth           0.2598      0.101      2.581      0.012       0.060
     0.460
carheight         -0.1763      0.053     -3.341      0.001      -0.281
    -0.071
curbweight         0.3292      0.131      2.521      0.014       0.070
     0.589
enginesize         2.0104      0.464      4.333      0.000       1.088
     2.933
boreratio         -0.6215      0.170     -3.660      0.000      -0.959
    -0.284
stroke            -0.1813      0.083     -2.174      0.032      -0.347
    -0.016
compressionratio  -0.3612      0.271     -1.334      0.186      -0.900
     0.177
horsepower        -0.2005      0.209     -0.958      0.341      -0.617
     0.216
peakrpm            0.1878      0.051      3.688      0.000       0.087
     0.289
citympg           -0.0516      0.154     -0.335      0.738      -0.357
     0.254
highwaympg         0.1273      0.139      0.918      0.361      -0.148
```

```
     0.403
CarName_audi            0.0697      0.086      0.812      0.419      -0.101
     0.240
CarName_bmw             0.3396      0.093      3.664      0.000       0.155
     0.524
CarName_buick           0.0493      0.089      0.554      0.581      -0.128
     0.226
CarName_chevrolet      -0.0522      0.079     -0.665      0.508      -0.208
     0.104
CarName_dodge          -0.1090      0.066     -1.655      0.102      -0.240
     0.022
CarName_honda          -0.0689      0.082     -0.836      0.406      -0.233
     0.095
CarName_isuzu           0.0023      0.074      0.031      0.975      -0.145
     0.149
CarName_jaguar         -0.1332      0.089     -1.492      0.139      -0.311
     0.044
CarName_mazda           0.0245      0.066      0.370      0.712      -0.107
     0.156
CarName_mercury     -5.682e-15   1.99e-15     -2.850      0.005   -9.65e-15
-1.72e-15
CarName_mitsubishi     -0.1273      0.067     -1.899      0.061      -0.261
     0.006
CarName_nissan          0.0268      0.068      0.396      0.693      -0.108
     0.162
CarName_peugeot        -0.2884      0.061     -4.719      0.000      -0.410
    -0.167
CarName_plymouth       -0.1125      0.064     -1.758      0.082      -0.240
     0.015
CarName_porsche         0.2637      0.111      2.367      0.020       0.042
     0.485
CarName_renault        -0.0109      0.079     -0.138      0.891      -0.168
     0.146
CarName_saab            0.2259      0.087      2.608      0.011       0.054
     0.398
CarName_subaru         -0.0403      0.108     -0.373      0.710      -0.255
     0.175
CarName_toyota          0.0165      0.063      0.263      0.794      -0.108
     0.141
CarName_volkswagen      0.0229      0.064      0.356      0.723      -0.105
     0.151
CarName_volvo           0.1069      0.092      1.164      0.247      -0.076
     0.289
carbody_hardtop        -0.0971      0.057     -1.704      0.092      -0.210
     0.016
carbody_hatchback      -0.1058      0.044     -2.386      0.019      -0.194
    -0.018
carbody_sedan          -0.0866      0.047     -1.838      0.070      -0.180
     0.007
carbody_wagon          -0.0676      0.050     -1.343      0.183      -0.168
     0.032
drivewheel_fwd         -0.0034      0.025     -0.140      0.889      -0.052
     0.045
drivewheel_rwd          0.0262      0.035      0.757      0.451      -0.043
     0.095
enginetype_dohcv        0.2379      0.168      1.415      0.161      -0.096
     0.572
```

```
enginetype_l             0.2071      0.082      2.528      0.013       0.044
     0.370
enginetype_ohc           0.0010      0.045      0.023      0.981      -0.088
     0.090
enginetype_ohcf          0.1731      0.052      3.324      0.001       0.070
     0.277
enginetype_ohcv         -0.0275      0.039     -0.702      0.485      -0.106
     0.050
enginetype_rotor         0.3961      0.115      3.438      0.001       0.167
     0.625
cylindernumber_five      0.2666      0.141      1.885      0.063      -0.015
     0.548
cylindernumber_four      0.4386      0.183      2.391      0.019       0.074
     0.803
cylindernumber_six       0.1189      0.103      1.154      0.252      -0.086
     0.324
cylindernumber_three     0.4955      0.124      3.992      0.000       0.249
     0.742
cylindernumber_twelve   -0.3563      0.179     -1.992      0.050      -0.712
    -0.001
cylindernumber_two       0.3961      0.115      3.438      0.001       0.167
     0.625
fuelsystem_2bbl          0.0232      0.056      0.416      0.679      -0.088
     0.134
fuelsystem_4bbl         -0.0488      0.079     -0.616      0.540      -0.206
     0.109
fuelsystem_idi           0.1491      0.163      0.915      0.363      -0.175
     0.473
fuelsystem_mfi                0          0        nan        nan           0
         0
fuelsystem_mpfi         -0.0161      0.062     -0.262      0.794      -0.138
     0.106
fuelsystem_spdi         -0.0262      0.066     -0.399      0.691      -0.157
     0.104
fuelsystem_spfi               0          0        nan        nan           0
         0
==============================================================================
Omnibus:                       34.574   Durbin-Watson:                   1.863
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              162.755
Skew:                           0.707   Prob(JB):                     4.55e-36
Kurtosis:                       8.032   Cond. No.                     1.07e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The smallest eigenvalue is 1.07e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

As the warnings suggest there is a strong multicolinearity problem.

Checking VIF

Variance Inflation Factor or VIF, gives a basic quantitative idea about how much the feature variables are correlated with each other. It is an extremely important parameter to test our linear model. The formula for calculating `VIF` is:

$$VIF_i = \frac{1}{1 - {R_i}^2}$$

In [46]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [47]:

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

```
                 Features         VIF
54     cylindernumber_three         inf
59           fuelsystem_idi         inf
35           CarName_subaru         inf
1                  fueltype         inf
30          CarName_peugeot         inf
46             enginetype_l         inf
48          enginetype_ohcf         inf
56       cylindernumber_two         inf
50         enginetype_rotor         inf
4            enginelocation         inf
52      cylindernumber_four  470.310000
10               enginesize  372.550000
13         compressionratio  298.700000
11                boreratio   89.540000
53       cylindernumber_six   89.140000
14               horsepower   86.860000
51      cylindernumber_five   86.020000
61          fuelsystem_mpfi   68.000000
16                  citympg   59.320000
9                curbweight   55.090000
57          fuelsystem_2bbl   51.750000
17               highwaympg   49.470000
41            carbody_sedan   40.420000
36           CarName_toyota   34.580000
40        carbody_hatchback   32.310000
47           enginetype_ohc   31.320000
6                 carlength   30.550000
29           CarName_nissan   29.560000
23            CarName_honda   29.190000
5                 wheelbase   26.560000
..                      ...         ...
42            carbody_wagon   21.290000
44           drivewheel_rwd   19.930000
20            CarName_buick   19.500000
28       CarName_mitsubishi   19.320000
32          CarName_porsche   18.580000
18             CarName_audi   18.110000
55    cylindernumber_twelve   16.190000
45         enginetype_dohcv   14.300000
37       CarName_volkswagen   14.020000
22            CarName_dodge   12.710000
62          fuelsystem_spdi   12.640000
12                   stroke   12.540000
25           CarName_jaguar   11.920000
34             CarName_saab   11.240000
43           drivewheel_fwd   10.580000
58          fuelsystem_4bbl    9.410000
```

```
8                carheight    9.350000
2               aspiration    9.250000
21        CarName_chevrolet    9.240000
24            CarName_isuzu    8.190000
31         CarName_plymouth    8.110000
15                  peakrpm    7.240000
49           enginetype_ohcv    6.610000
33          CarName_renault    6.250000
0                 symboling    5.760000
39          carbody_hardtop    4.870000
3               doornumber    4.110000
27          CarName_mercury         NaN
60           fuelsystem_mfi         NaN
63          fuelsystem_spfi         NaN

[64 rows x 2 columns]
```

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values some with 'inf'. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `fueltype` has high VIF inf and a high p-value (`0.206`) as well. Hence, this variable isn't of much use and should be dropped.

In [48]:
```
X = X_train.drop('fueltype', 1,)
```

In [49]:
```
# Build a second fitted model
X_train_lm = sm.add_constant(X)

lr_2 = sm.OLS(y_train, X_train_lm).fit()
# Print the summary of the model
print(lr_2.summary())
```

```
49
Time:                        07:35:48   Log-Likelihood:                 280.
02
No. Observations:                 143   AIC:                            -44
4.0
Df Residuals:                      85   BIC:                            -27
2.2
Df Model:                          57

Covariance Type:            nonrobust

=============================================================================
=============
                            coef    std err          t      P>|t|      [0.02
5      0.975]
-----------------------------------------------------------------------------
-------------
const                    -0.0715      0.169     -0.423      0.673      -0.40
7       0.264
symboling                -0.0099      0.037     -0.265      0.792      -0.08
```

In [50]:
```
# Calculate the VIFs again for the new model

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[50]:

	Features	VIF
49	enginetype_rotor	inf
45	enginetype_l	inf
29	CarName_peugeot	inf
53	cylindernumber_three	inf
55	cylindernumber_two	inf
3	enginelocation	2631.800000
51	cylindernumber_four	470.310000
34	CarName_subaru	447.870000
9	enginesize	372.550000
58	fuelsystem_idi	317.620000
12	compressionratio	298.700000
47	enginetype_ohcf	287.690000
10	boreratio	89.540000
52	cylindernumber_six	89.140000
13	horsepower	86.860000
50	cylindernumber_five	86.020000
60	fuelsystem_mpfi	68.000000
15	citympg	59.320000
8	curbweight	55.090000
56	fuelsystem_2bbl	51.750000
16	highwaympg	49.470000
40	carbody_sedan	40.420000
35	CarName_toyota	34.580000
39	carbody_hatchback	32.310000
46	enginetype_ohc	31.320000
5	carlength	30.550000
28	CarName_nissan	29.560000
22	CarName_honda	29.190000
4	wheelbase	26.560000

	Features	VIF
25	CarName_mazda	26.350000
...	...	...
41	carbody_wagon	21.290000
43	drivewheel_rwd	19.930000
19	CarName_buick	19.500000
27	CarName_mitsubishi	19.320000
31	CarName_porsche	18.580000
17	CarName_audi	18.110000
54	cylindernumber_twelve	16.190000
44	enginetype_dohcv	14.300000
36	CarName_volkswagen	14.020000
21	CarName_dodge	12.710000
61	fuelsystem_spdi	12.640000
11	stroke	12.540000
24	CarName_jaguar	11.920000
33	CarName_saab	11.240000
42	drivewheel_fwd	10.580000
57	fuelsystem_4bbl	9.410000
7	carheight	9.350000
1	aspiration	9.250000
20	CarName_chevrolet	9.240000
23	CarName_isuzu	8.190000
30	CarName_plymouth	8.110000
14	peakrpm	7.240000
48	enginetype_ohcv	6.610000
32	CarName_renault	6.250000
0	symboling	5.760000
38	carbody_hardtop	4.870000
2	doornumber	4.110000
26	CarName_mercury	NaN
59	fuelsystem_mfi	NaN
62	fuelsystem_spfi	NaN

63 rows × 2 columns

In [51]:
```
# Calculate the VIFs again for the new model

X1 = X.drop('enginetype_rotor', 1,)
vif = pd.DataFrame()
vif['Features'] = X1.columns
vif['VIF'] = [variance_inflation_factor(X1.values, i) for i in range(X1.shape[1])
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[51]:

	Features	VIF
52	cylindernumber_three	inf
29	CarName_peugeot	inf
45	enginetype_l	inf
3	enginelocation	2631.800000
50	cylindernumber_four	470.310000
34	CarName_subaru	447.870000
9	enginesize	372.550000
57	fuelsystem_idi	317.620000
12	compressionratio	298.700000
47	enginetype_ohcf	287.690000
54	cylindernumber_two	105.250000

In [52]:
```
# Calculate the VIFs again for the new model

X2 = X1.drop('cylindernumber_three', 1,)
vif = pd.DataFrame()
vif['Features'] = X2.columns
vif['VIF'] = [variance_inflation_factor(X2.values, i) for i in range(X2.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[52]:

	Features	VIF
3	enginelocation	2631.80
50	cylindernumber_four	470.31
34	CarName_subaru	447.87
9	enginesize	372.55
56	fuelsystem_idi	317.62
12	compressionratio	298.70
47	enginetype_ohcf	287.69
45	enginetype_l	174.12
29	CarName_peugeot	121.59
53	cylindernumber_two	105.25
10	boreratio	89.54

In [53]:
```
# Build a second fitted model
X_train_lm = sm.add_constant(X)

lr_2 = sm.OLS(y_train, X_train_lm).fit()
# Print the summary of the model
print(lr_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.975
Model:                            OLS   Adj. R-squared:                  0.958
Method:                 Least Squares   F-statistic:                     57.59
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           1.40e-49
Time:                        07:35:50   Log-Likelihood:                 280.02
No. Observations:                 143   AIC:                            -444.0
Df Residuals:                      85   BIC:                            -272.2
Df Model:                          57
Covariance Type:            nonrobust
================================================================================
==========
                           coef    std err          t      P>|t|      [0.025
    0.975]
--------------------------------------------------------------------------------
----------
const                   -0.0715      0.169     -0.423      0.673      -0.407
     0.264
symboling               -0.0099      0.037     -0.265      0.792      -0.084
     0.064
aspiration              -0.0811      0.029     -2.778      0.007      -0.139
    -0.023
doornumber              -0.0127      0.015     -0.841      0.403      -0.043
     0.017
enginelocation          -0.2218      0.056     -3.988      0.000      -0.332
    -0.111
wheelbase                0.2368      0.093      2.542      0.013       0.052
     0.422
carlength               -0.1752      0.100     -1.747      0.084      -0.375
     0.024
carwidth                 0.2598      0.101      2.581      0.012       0.060
     0.460
carheight               -0.1763      0.053     -3.341      0.001      -0.281
    -0.071
curbweight               0.3292      0.131      2.521      0.014       0.070
     0.589
enginesize               2.0104      0.464      4.333      0.000       1.088
     2.933
boreratio               -0.6215      0.170     -3.660      0.000      -0.959
    -0.284
stroke                  -0.1813      0.083     -2.174      0.032      -0.347
    -0.016
compressionratio        -0.3612      0.271     -1.334      0.186      -0.900
     0.177
horsepower              -0.2005      0.209     -0.958      0.341      -0.617
     0.216
peakrpm                  0.1878      0.051      3.688      0.000       0.087
     0.289
```

```
citympg              -0.0516      0.154     -0.335      0.738      -0.357
     0.254
highwaympg            0.1273      0.139      0.918      0.361      -0.148
     0.403
CarName_audi          0.0697      0.086      0.812      0.419      -0.101
     0.240
CarName_bmw           0.3396      0.093      3.664      0.000       0.155
     0.524
CarName_buick         0.0493      0.089      0.554      0.581      -0.128
     0.226
CarName_chevrolet    -0.0522      0.079     -0.665      0.508      -0.208
     0.104
CarName_dodge        -0.1090      0.066     -1.655      0.102      -0.240
     0.022
CarName_honda        -0.0689      0.082     -0.836      0.406      -0.233
     0.095
CarName_isuzu         0.0023      0.074      0.031      0.975      -0.145
     0.149
CarName_jaguar       -0.1332      0.089     -1.492      0.139      -0.311
     0.044
CarName_mazda         0.0245      0.066      0.370      0.712      -0.107
     0.156
CarName_mercury    3.465e-15   1.05e-15      3.299      0.001    1.38e-15
  5.55e-15
CarName_mitsubishi   -0.1273      0.067     -1.899      0.061      -0.261
     0.006
CarName_nissan        0.0268      0.068      0.396      0.693      -0.108
     0.162
CarName_peugeot      -0.2884      0.061     -4.719      0.000      -0.410
    -0.167
CarName_plymouth     -0.1125      0.064     -1.758      0.082      -0.240
     0.015
CarName_porsche       0.2637      0.111      2.367      0.020       0.042
     0.485
CarName_renault      -0.0109      0.079     -0.138      0.891      -0.168
     0.146
CarName_saab          0.2259      0.087      2.608      0.011       0.054
     0.398
CarName_subaru       -0.0087      0.118     -0.074      0.941      -0.243
     0.226
CarName_toyota        0.0165      0.063      0.263      0.794      -0.108
     0.141
CarName_volkswagen    0.0229      0.064      0.356      0.723      -0.105
     0.151
CarName_volvo         0.1069      0.092      1.164      0.247      -0.076
     0.289
carbody_hardtop      -0.0971      0.057     -1.704      0.092      -0.210
     0.016
carbody_hatchback    -0.1058      0.044     -2.386      0.019      -0.194
    -0.018
carbody_sedan        -0.0866      0.047     -1.838      0.070      -0.180
     0.007
carbody_wagon        -0.0676      0.050     -1.343      0.183      -0.168
     0.032
drivewheel_fwd       -0.0034      0.025     -0.140      0.889      -0.052
     0.045
drivewheel_rwd        0.0262      0.035      0.757      0.451      -0.043
```

```
     0.095
enginetype_dohcv          0.2379      0.168      1.415      0.161      -0.096
     0.572
enginetype_l              0.2071      0.082      2.528      0.013       0.044
     0.370
enginetype_ohc            0.0010      0.045      0.023      0.981      -0.088
     0.090
enginetype_ohcf           0.1416      0.059      2.393      0.019       0.024
     0.259
enginetype_ohcv          -0.0275      0.039     -0.702      0.485      -0.106
     0.050
enginetype_rotor          0.3961      0.115      3.438      0.001       0.167
     0.625
cylindernumber_five       0.2666      0.141      1.885      0.063      -0.015
     0.548
cylindernumber_four       0.4386      0.183      2.391      0.019       0.074
     0.803
cylindernumber_six        0.1189      0.103      1.154      0.252      -0.086
     0.324
cylindernumber_three      0.4955      0.124      3.992      0.000       0.249
     0.742
cylindernumber_twelve    -0.3563      0.179     -1.992      0.050      -0.712
    -0.001
cylindernumber_two        0.3961      0.115      3.438      0.001       0.167
     0.625
fuelsystem_2bbl           0.0232      0.056      0.416      0.679      -0.088
     0.134
fuelsystem_4bbl          -0.0488      0.079     -0.616      0.540      -0.206
     0.109
fuelsystem_idi            0.2752      0.230      1.198      0.234      -0.181
     0.732
fuelsystem_mfi                 0          0        nan        nan           0
         0
fuelsystem_mpfi          -0.0161      0.062     -0.262      0.794      -0.138
     0.106
fuelsystem_spdi          -0.0262      0.066     -0.399      0.691      -0.157
     0.104
fuelsystem_spfi                0          0        nan        nan           0
         0
==============================================================================
Omnibus:                       34.574   Durbin-Watson:                   1.863
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              162.755
Skew:                           0.707   Prob(JB):                     4.55e-36
Kurtosis:                       8.032   Cond. No.                     1.08e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The smallest eigenvalue is 9.47e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

After droping two features we see VIF to be reduced and p value change as well, But the manual feature list selection will not not help due to the number of features lets move to

RFE

In [96]:
```
from sklearn.model_selection import train_test_split

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

In [97]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
# Apply scaler() to all the columns except the 'yes-no' and 'dummy' variables
num_vars = ['symboling','wheelbase','carlength','carwidth','carheight','curbweigh
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

In [98]:
```
df_train.head()
```

Out[98]:

	symboling	fueltype	aspiration	doornumber	enginelocation	wheelbase	carlength	carwidt
car_ID								
123	0.6	1	1	0	1	0.244828	0.426016	0.29166
126	1.0	1	1	1	1	0.272414	0.452033	0.66666
167	0.6	1	1	1	1	0.272414	0.448780	0.30833
2	1.0	1	1	1	1	0.068966	0.450407	0.31666
200	0.2	1	0	0	1	0.610345	0.775610	0.57500

5 rows × 65 columns

In [99]:
```
df_train.describe()
```

Out[99]:

	symboling	fueltype	aspiration	doornumber	enginelocation	wheelbase	carlength	
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	14
mean	0.559441	0.909091	0.818182	0.440559	0.993007	0.411141	0.525476	
std	0.239200	0.288490	0.387050	0.498199	0.083624	0.205581	0.204848	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.400000	1.000000	1.000000	0.000000	1.000000	0.272414	0.399187	
50%	0.600000	1.000000	1.000000	0.000000	1.000000	0.341379	0.502439	
75%	0.600000	1.000000	1.000000	1.000000	1.000000	0.503448	0.669919	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 65 columns

In [100]:
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
```

In [59]:
```
# Running RFE with the output number of the variable as 10
lm = LinearRegression()
lm.fit(X_train,y_train)

rfe = RFE(lm,10)
rfe = rfe.fit(X_train,y_train)
```

In [60]:
```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

```
 ('boreratio', True, 1),
 ('stroke', False, 2),
 ('compressionratio', False, 25),
 ('horsepower', False, 46),
 ('peakrpm', False, 19),
 ('citympg', False, 38),
 ('highwaympg', False, 27),
 ('CarName_audi', False, 37),
 ('CarName_bmw', True, 1),
 ('CarName_buick', False, 36),
 ('CarName_chevrolet', False, 22),
 ('CarName_dodge', False, 18),
 ('CarName_honda', False, 21),
 ('CarName_isuzu', False, 53),
 ('CarName_jaguar', False, 32),
 ('CarName_mazda', False, 44),
 ('CarName_mercury', False, 48),
 ('CarName_mitsubishi', False, 13),
 ('CarName_nissan', False, 43),
 ('CarName_peugeot', False, 6),
```

In [61]:
```
col = X_train.columns[rfe.support_]
col
```

Out[61]:
```
Index(['enginelocation', 'carwidth', 'curbweight', 'enginesize', 'boreratio',
       'CarName_bmw', 'CarName_porsche', 'cylindernumber_three',
       'cylindernumber_twelve', 'cylindernumber_two'],
      dtype='object')
```

In [62]:
```
X_train.columns[~rfe.support_]
```

Out[62]:
```
Index(['symboling', 'fueltype', 'aspiration', 'doornumber', 'wheelbase',
       'carlength', 'carheight', 'stroke', 'compressionratio', 'horsepower',
       'peakrpm', 'citympg', 'highwaympg', 'CarName_audi', 'CarName_buick',
       'CarName_chevrolet', 'CarName_dodge', 'CarName_honda', 'CarName_isuzu',
       'CarName_jaguar', 'CarName_mazda', 'CarName_mercury',
       'CarName_mitsubishi', 'CarName_nissan', 'CarName_peugeot',
       'CarName_plymouth', 'CarName_renault', 'CarName_saab', 'CarName_subaru',
       'CarName_toyota', 'CarName_volkswagen', 'CarName_volvo',
       'carbody_hardtop', 'carbody_hatchback', 'carbody_sedan',
       'carbody_wagon', 'drivewheel_fwd', 'drivewheel_rwd', 'enginetype_dohcv',
       'enginetype_l', 'enginetype_ohc', 'enginetype_ohcf', 'enginetype_ohcv',
       'enginetype_rotor', 'cylindernumber_five', 'cylindernumber_four',
       'cylindernumber_six', 'fuelsystem_2bbl', 'fuelsystem_4bbl',
       'fuelsystem_idi', 'fuelsystem_mfi', 'fuelsystem_mpfi',
       'fuelsystem_spdi', 'fuelsystem_spfi'],
      dtype='object')
```

In [63]:
```
x_train_rfe=X_train[col]
```

In [64]:
```
import statsmodels.api as sm
```

In [65]:
```
x_train_rfe = sm.add_constant(x_train_rfe)
lm = sm.OLS(y_train,x_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.912
Model:                            OLS   Adj. R-squared:                  0.906
Method:                 Least Squares   F-statistic:                     137.1
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           1.21e-64
Time:                        07:35:52   Log-Likelihood:                 190.87
No. Observations:                 143   AIC:                            -359.7
Df Residuals:                     132   BIC:                            -327.1
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
==========
                           coef    std err          t      P>|t|      [0.025
    0.975]
--------------------------------------------------------------------------------
----------
const                    0.1704      0.084      2.022      0.045       0.004
     0.337
enginelocation          -0.3298      0.086     -3.844      0.000      -0.500
    -0.160
carwidth                 0.3283      0.067      4.910      0.000       0.196
     0.461
curbweight               0.2989      0.080      3.729      0.000       0.140
     0.457
enginesize               0.5732      0.097      5.926      0.000       0.382
     0.764
boreratio               -0.1088      0.037     -2.976      0.003      -0.181
    -0.036
CarName_bmw              0.2453      0.029      8.327      0.000       0.187
     0.304
CarName_porsche          0.1565      0.052      3.000      0.003       0.053
     0.260
cylindernumber_three     0.1802      0.068      2.638      0.009       0.045
     0.315
cylindernumber_twelve   -0.0536      0.081     -0.659      0.511      -0.214
     0.107
cylindernumber_two       0.1542      0.037      4.123      0.000       0.080
     0.228
==============================================================================
Omnibus:                       10.995   Durbin-Watson:                   1.961
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               15.683
Skew:                           0.423   Prob(JB):                     0.000393
Kurtosis:                       4.385   Cond. No.                         38.7
==============================================================================

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

Calculate VIF1

In [66]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [67]:
```
vif = pd.DataFrame()
x = x_train_rfe
vif['features']=x.columns
vif['vif']=[variance_inflation_factor(x.values, i ) for i in range(x.shape[1])]
vif
```

Out[67]:

	features	vif
0	const	230.922520
1	enginelocation	1.663007
2	carwidth	4.920153
3	curbweight	9.263420
4	enginesize	7.225349
5	boreratio	1.852666
6	CarName_bmw	1.134883
7	CarName_porsche	1.818873
8	cylindernumber_three	1.054182
9	cylindernumber_twelve	1.494303
10	cylindernumber_two	1.237541

In [68]:
```
# Running RFE with the output number of the variable as 12
lm = LinearRegression()
lm.fit(X_train,y_train)

rfe = RFE(lm,12)
rfe = rfe.fit(X_train,y_train)
```

In [69]:
```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[69]:
```
[('symboling', False, 39),
 ('fueltype', False, 21),
 ('aspiration', False, 12),
 ('doornumber', False, 49),
 ('enginelocation', True, 1),
 ('wheelbase', False, 13),
 ('carlength', False, 18),
 ('carwidth', True, 1),
 ('carheight', False, 14),
 ('curbweight', True, 1),
 ('enginesize', True, 1),
 ('boreratio', True, 1),
 ('stroke', True, 1),
 ('compressionratio', False, 23),
 ('horsepower', False, 44),
 ('peakrpm', False, 17),
 ('citympg', False, 36),
 ('highwaympg', False, 25),
 ('CarName_audi', False, 35),
 ('CarName_bmw', True, 1),
 ('CarName_buick', False, 34),
 ('CarName_chevrolet', False, 20),
 ('CarName_dodge', False, 16),
 ('CarName_honda', False, 19),
 ('CarName_isuzu', False, 51),
 ('CarName_jaguar', False, 30),
 ('CarName_mazda', False, 42),
 ('CarName_mercury', False, 46),
 ('CarName_mitsubishi', False, 11),
 ('CarName_nissan', False, 41),
 ('CarName_peugeot', False, 4),
 ('CarName_plymouth', False, 15),
 ('CarName_porsche', True, 1),
 ('CarName_renault', False, 45),
 ('CarName_saab', False, 7),
 ('CarName_subaru', False, 9),
 ('CarName_toyota', False, 43),
 ('CarName_volkswagen', False, 40),
 ('CarName_volvo', False, 6),
 ('carbody_hardtop', False, 26),
 ('carbody_hatchback', False, 24),
 ('carbody_sedan', False, 27),
 ('carbody_wagon', False, 28),
 ('drivewheel_fwd', False, 48),
 ('drivewheel_rwd', False, 38),
 ('enginetype_dohcv', False, 10),
 ('enginetype_l', False, 5),
 ('enginetype_ohc', False, 50),
 ('enginetype_ohcf', False, 8),
 ('enginetype_ohcv', False, 37),
 ('enginetype_rotor', True, 1),
 ('cylindernumber_five', False, 3),
 ('cylindernumber_four', False, 2),
 ('cylindernumber_six', False, 33),
```

```
 ('cylindernumber_three', True, 1),
 ('cylindernumber_twelve', True, 1),
 ('cylindernumber_two', True, 1),
 ('fuelsystem_2bbl', False, 47),
 ('fuelsystem_4bbl', False, 31),
 ('fuelsystem_idi', False, 22),
 ('fuelsystem_mfi', False, 52),
 ('fuelsystem_mpfi', False, 32),
 ('fuelsystem_spdi', False, 29),
 ('fuelsystem_spfi', False, 53)]
```

In [70]:
```
col = X_train.columns[rfe.support_]
print(col)
x_train_rfe=X_train[col]
x_train_rfe = sm.add_constant(x_train_rfe)
lm = sm.OLS(y_train,x_train_rfe).fit()
print(lm.summary())
vif = pd.DataFrame()
x = x_train_rfe
vif['features']=x.columns
vif['vif']=[variance_inflation_factor(x.values, i ) for i in range(x.shape[1])]
print(vif)
```

```
Index(['enginelocation', 'carwidth', 'curbweight', 'enginesize', 'boreratio',
       'stroke', 'CarName_bmw', 'CarName_porsche', 'enginetype_rotor',
       'cylindernumber_three', 'cylindernumber_twelve', 'cylindernumber_two'],
      dtype='object')
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.916
Model:                            OLS   Adj. R-squared:                  0.909
Method:                 Least Squares   F-statistic:                     130.2
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           6.86e-65
Time:                        07:35:53   Log-Likelihood:                 194.22
No. Observations:                 143   AIC:                            -364.4
Df Residuals:                     131   BIC:                            -328.9
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
==========
                           coef    std err          t      P>|t|      [0.025
    0.975]
--------------------------------------------------------------------------------
----------
const                    0.2085      0.084      2.483      0.014       0.042
     0.375
enginelocation          -0.3126      0.084     -3.704      0.000      -0.480
    -0.146
carwidth                 0.3477      0.066      5.266      0.000       0.217
     0.478
curbweight               0.2791      0.079      3.534      0.001       0.123
     0.435
enginesize               0.6626      0.101      6.541      0.000       0.462
     0.863
boreratio               -0.1483      0.039     -3.788      0.000      -0.226
    -0.071
stroke                  -0.1067      0.043     -2.510      0.013      -0.191
    -0.023
CarName_bmw              0.2323      0.029      7.917      0.000       0.174
     0.290
CarName_porsche          0.1472      0.051      2.869      0.005       0.046
     0.249
enginetype_rotor         0.0855      0.019      4.589      0.000       0.049
     0.122
cylindernumber_three     0.1773      0.067      2.647      0.009       0.045
     0.310
cylindernumber_twelve   -0.1384      0.087     -1.598      0.112      -0.310
```

```
     0.033
cylindernumber_two        0.0855      0.019      4.589      0.000       0.049
     0.122
==============================================================================
Omnibus:                       13.877   Durbin-Watson:                   2.071
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               24.547
Skew:                           0.444   Prob(JB):                     4.67e-06
Kurtosis:                       4.825   Cond. No.                     9.27e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The smallest eigenvalue is 4.99e-32. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
                 features         vif
0                   const  238.746750
1          enginelocation    1.674027
2                carwidth    4.988429
3              curbweight    9.356507
4              enginesize    8.244360
5               boreratio    2.210802
6                  stroke    1.512427
7             CarName_bmw    1.171284
8         CarName_porsche    1.828551
9        enginetype_rotor         inf
10   cylindernumber_three    1.054490
11  cylindernumber_twelve    1.762497
12     cylindernumber_two         inf
```

In [71]:

```
# Running RFE with the output number of the variable as 8
lm = LinearRegression()
lm.fit(X_train,y_train)

rfe = RFE(lm,6)
rfe = rfe.fit(X_train,y_train)
```

In [72]:
```
print(list(zip(X_train.columns,rfe.support_,rfe.ranking_)))
col = X_train.columns[rfe.support_]
print(col)
x_train_rfe=X_train[col]
x_train_rfe = sm.add_constant(x_train_rfe)
lm = sm.OLS(y_train,x_train_rfe).fit()
print(lm.summary())
vif = pd.DataFrame()
x = x_train_rfe
vif['features']=x.columns
vif['vif']=[variance_inflation_factor(x.values, i ) for i in range(x.shape[1])]
print(vif)
```

```
[('symboling', False, 45), ('fueltype', False, 27), ('aspiration', False, 18),
('doornumber', False, 55), ('enginelocation', True, 1), ('wheelbase', False, 1
9), ('carlength', False, 24), ('carwidth', True, 1), ('carheight', False, 20),
('curbweight', True, 1), ('enginesize', True, 1), ('boreratio', False, 4), ('st
roke', False, 6), ('compressionratio', False, 29), ('horsepower', False, 50),
('peakrpm', False, 23), ('citympg', False, 42), ('highwaympg', False, 31), ('Ca
rName_audi', False, 41), ('CarName_bmw', True, 1), ('CarName_buick', False, 4
0), ('CarName_chevrolet', False, 26), ('CarName_dodge', False, 22), ('CarName_h
onda', False, 25), ('CarName_isuzu', False, 57), ('CarName_jaguar', False, 36),
('CarName_mazda', False, 48), ('CarName_mercury', False, 52), ('CarName_mitsubi
shi', False, 17), ('CarName_nissan', False, 47), ('CarName_peugeot', False, 1
0), ('CarName_plymouth', False, 21), ('CarName_porsche', False, 3), ('CarName_r
enault', False, 51), ('CarName_saab', False, 13), ('CarName_subaru', False, 1
5), ('CarName_toyota', False, 49), ('CarName_volkswagen', False, 46), ('CarName
_volvo', False, 12), ('carbody_hardtop', False, 32), ('carbody_hatchback', Fals
e, 30), ('carbody_sedan', False, 33), ('carbody_wagon', False, 34), ('drivewhee
l_fwd', False, 54), ('drivewheel_rwd', False, 44), ('enginetype_dohcv', False,
16), ('enginetype_l', False, 11), ('enginetype_ohc', False, 56), ('enginetype_o
hcf', False, 14), ('enginetype_ohcv', False, 43), ('enginetype_rotor', False,
7), ('cylindernumber_five', False, 9), ('cylindernumber_four', False, 8), ('cyl
indernumber_six', False, 39), ('cylindernumber_three', True, 1), ('cylindernumb
er_twelve', False, 5), ('cylindernumber_two', False, 2), ('fuelsystem_2bbl', Fa
lse, 53), ('fuelsystem_4bbl', False, 37), ('fuelsystem_idi', False, 28), ('fuel
system_mfi', False, 58), ('fuelsystem_mpfi', False, 38), ('fuelsystem_spdi', Fa
lse, 35), ('fuelsystem_spfi', False, 59)]
Index(['enginelocation', 'carwidth', 'curbweight', 'enginesize', 'CarName_bmw',
       'cylindernumber_three'],
      dtype='object')
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.892
Model:                            OLS   Adj. R-squared:                  0.887
Method:                 Least Squares   F-statistic:                     186.8
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           4.11e-63
Time:                        07:35:53   Log-Likelihood:                 175.94
No. Observations:                 143   AIC:                            -337.9
Df Residuals:                     136   BIC:                            -317.1
Df Model:                           6
Covariance Type:            nonrobust
===============================================================================
=========
                           coef    std err          t      P>|t|      [0.025
   0.975]
```

```
------------------------------------------------------------------------------
---------
const                     0.2907      0.076      3.841      0.000       0.141
    0.440
enginelocation           -0.4824      0.076     -6.379      0.000      -0.632
   -0.333
carwidth                  0.3866      0.070      5.532      0.000       0.248
    0.525
curbweight                0.2679      0.078      3.451      0.001       0.114
    0.421
enginesize                0.4480      0.082      5.460      0.000       0.286
    0.610
CarName_bmw               0.2497      0.032      7.917      0.000       0.187
    0.312
cylindernumber_three      0.1926      0.075      2.584      0.011       0.045
    0.340
==============================================================================
Omnibus:                        4.900   Durbin-Watson:                   2.031
Prob(Omnibus):                  0.086   Jarque-Bera (JB):                4.619
Skew:                           0.321   Prob(JB):                       0.0993
Kurtosis:                       3.603   Cond. No.                         29.2
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
               features         vif
0                 const  155.842927
1        enginelocation    1.080674
2              carwidth    4.490868
3            curbweight    7.265199
4            enginesize    4.348564
5           CarName_bmw    1.088127
6  cylindernumber_three    1.049848
```

Step 5: Residual Analysis of the train data

So, now to check if the error terms are also normally distributed (which is infact, one of the major assumptions of linear regression), let us plot the histogram of the error terms and see what it looks like.

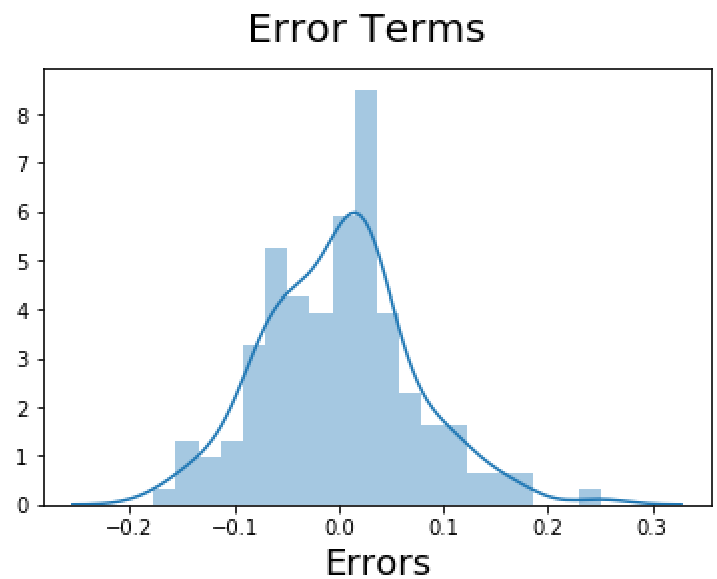
In [78]:

```
# Build a fourth fitted model
x_train_rfe=X_train[col]
X_train_lm = sm.add_constant(x_train_rfe)

lr_4 = sm.OLS(y_train, X_train_lm).fit()
y_train_price = lr_4.predict(X_train_lm)
```

In [79]:
```
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)                  # Plot heading
plt.xlabel('Errors', fontsize = 18)                         # X-label
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserW
arning: The 'normed' kwarg is deprecated, and has been replaced by the 'densit
y' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[79]: Text(0.5,0,'Errors')



Step 6: Making Predictions Using the Final Model

Now that we have fitted the model and checked the normality of error terms, it's time to go ahead and make predictions using the final, i.e. fourth model.

Applying the scaling on the test sets

In [101]:

```
#x_train_rfe=X_train[col]
from sklearn.model_selection import train_test_split

# We specify this so that the train and test data set always have the same rows, 
np.random.seed(0)
df_train, df_test = train_test_split(GeelyAuto, train_size = 0.7, test_size = 0.3
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
# Apply scaler() to all the columns except the 'yes-no' and 'dummy' variables
num_vars = ['symboling','wheelbase','carlength','carwidth','carheight','curbweigh
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
num_vars = ['symboling','wheelbase','carlength','carwidth','carheight','curbweigh
df_test[num_vars] = scaler.transform(df_test[num_vars])
df_test.describe()
```

Out[101]:

	symboling	fueltype	aspiration	doornumber	enginelocation	wheelbase	carlength	carw
count	62.000000	62.000000	62.000000	62.000000	62.000000	62.000000	62.000000	62.000
mean	0.583871	0.887097	0.822581	0.435484	0.967742	0.437764	0.559481	0.480
std	0.271724	0.319058	0.385142	0.499868	0.178127	0.212861	0.189947	0.165
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.056911	0.183
25%	0.400000	1.000000	1.000000	0.000000	1.000000	0.313793	0.459350	0.358
50%	0.600000	1.000000	1.000000	0.000000	1.000000	0.387931	0.547967	0.441
75%	0.800000	1.000000	1.000000	1.000000	1.000000	0.570690	0.719919	0.516
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.182759	1.089431	0.975

8 rows × 65 columns

In [104]:

```
col = ['enginelocation', 'carwidth', 'curbweight', 'enginesize', 'CarName_bmw',
       'cylindernumber_three']
y_test = df_test.pop('price')
X_test = df_test[col]
y_train = df_train.pop('price')
X_train = df_train[col]
```

In [105]:

```
# Adding constant variable to test dataframe
X_test_final = sm.add_constant(X_test)
X_train_final = sm.add_constant(X_train)
```

In [106]:

```
lr_2 = sm.OLS(y_train, X_train_final).fit()
```

In [110]:
```
lr_2.params
```

Out[110]:
```
const                   0.290687
enginelocation         -0.482417
carwidth                0.386564
curbweight              0.267873
enginesize              0.447988
CarName_bmw             0.249720
cylindernumber_three    0.192642
dtype: float64
```

In [111]:
```
# Print the summary of the model
print(lr_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.892
Model:                            OLS   Adj. R-squared:                  0.887
Method:                 Least Squares   F-statistic:                     186.8
Date:                Sun, 11 Nov 2018   Prob (F-statistic):           4.11e-63
Time:                        10:30:01   Log-Likelihood:                 175.94
No. Observations:                 143   AIC:                            -337.9
Df Residuals:                     136   BIC:                            -317.1
Df Model:                           6
Covariance Type:            nonrobust
===============================================================================
=========
                           coef    std err          t      P>|t|      [0.025
   0.975]
-------------------------------------------------------------------------------
---------
const                    0.2907      0.076      3.841      0.000       0.141
    0.440
enginelocation          -0.4824      0.076     -6.379      0.000      -0.632
   -0.333
carwidth                 0.3866      0.070      5.532      0.000       0.248
    0.525
curbweight               0.2679      0.078      3.451      0.001       0.114
    0.421
enginesize               0.4480      0.082      5.460      0.000       0.286
    0.610
CarName_bmw              0.2497      0.032      7.917      0.000       0.187
    0.312
cylindernumber_three     0.1926      0.075      2.584      0.011       0.045
    0.340
==============================================================================
Omnibus:                        4.900   Durbin-Watson:                   2.031
Prob(Omnibus):                  0.086   Jarque-Bera (JB):                4.619
Skew:                           0.321   Prob(JB):                       0.0993
Kurtosis:                       3.603   Cond. No.                         29.2
==============================================================================

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

In [108]:
```
# Making predictions using the fourth model

y_pred_train_final = lr_2.predict(X_test_final)
y_pred_test_final = lr_2.predict(X_test_final)
```

In [113]:
```
# Check the parameters obtained

lr_2.params
```

Out[113]:
```
const                  0.290687
enginelocation        -0.482417
carwidth               0.386564
curbweight             0.267873
enginesize             0.447988
CarName_bmw            0.249720
cylindernumber_three   0.192642
dtype: float64
```

We can see that the equation of our best fitted line is: ¶

price=-0.4824×enginelocation+0.3865×carwidth+0.2678×curbweight+0.4479×enginesize+0.2497×Ca

In [ ]:
```

```