# 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.

# Reading data

In [88]:

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

In [114]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sn
```

In [90]:

```
cars=pd.read_csv("C:\CarPrice_Assignment.csv")
cars.head()
```

Out[90]:

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

5 rows × 26 columns

In [91]:

```
cars.shape
```

Out[91]:

```
(205, 26)
```

In [92]:

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

In [93]:

```
cars.describe()
```

Out[93]:

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

In [94]:

```
cars.columns
```

Out[94]:

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

In [95]:

```
cars.isnull()
```

Out[95]:

| | car_ID | symboling | CarName | fueltype | aspiration | doornumber | carbody | drivewheel | engi |
|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | |
| **1** | False | False | False | False | False | False | False | False | |
| **2** | False | False | False | False | False | False | False | False | |
| **3** | False | False | False | False | False | False | False | False | |
| **4** | False | False | False | False | False | False | False | False | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **200** | False | False | False | False | False | False | False | False | |
| **201** | False | False | False | False | False | False | False | False | |
| **202** | False | False | False | False | False | False | False | False | |
| **203** | False | False | False | False | False | False | False | False | |
| **204** | False | False | False | False | False | False | False | False | |

205 rows × 26 columns

In [96]:

```
cars.duplicated(subset = 'car_ID')== 0
```

Out[96]:

```
0      True
1      True
2      True
3      True
4      True
       ...
200    True
201    True
202    True
203    True
204    True
Length: 205, dtype: bool
```

In [97]:

```
cars.price.describe()
```

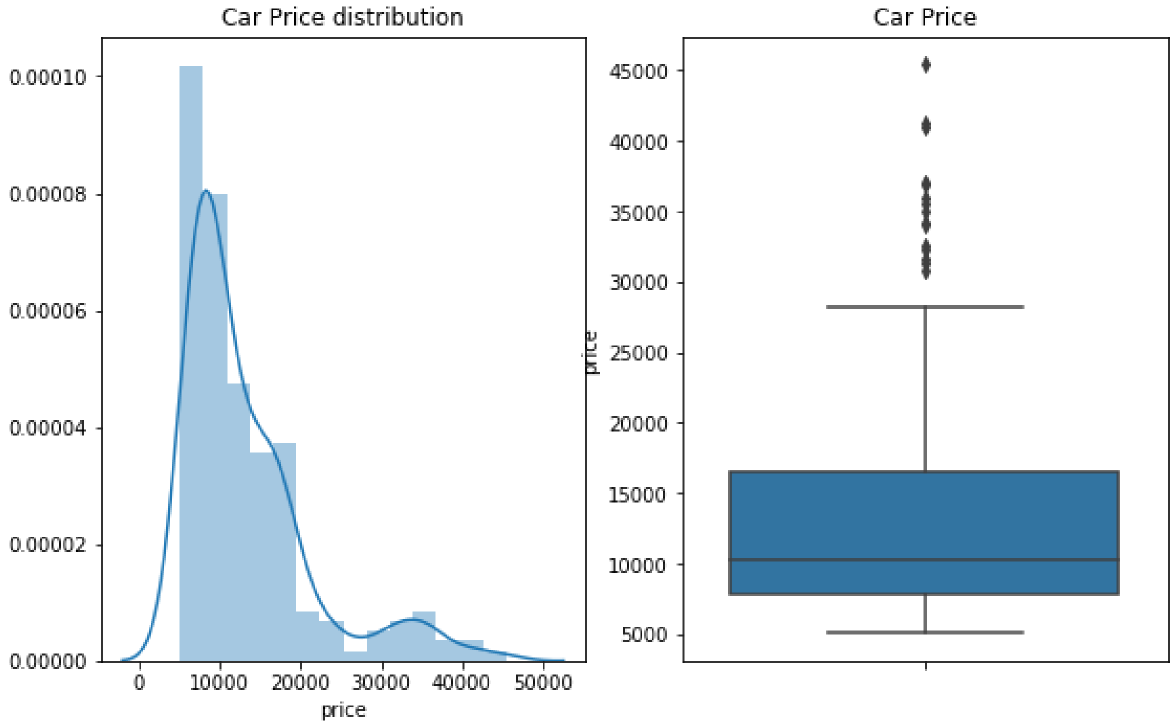
Out[97]:

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

# visualizing the data

In [98]:

```
plt.figure(figsize=(10,6))
plt.subplot(1,2,1)
plt.title('Car Price distribution')
sn.distplot(cars.price)
plt.subplot(1,2,2)
plt.title('Car Price')
sn.boxplot(y=cars.price)
plt.show()
```

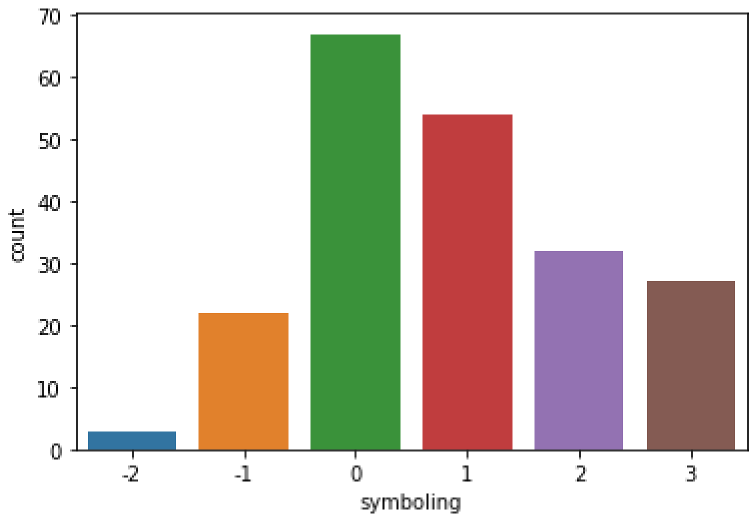



In [99]:

```
sn.countplot(cars['symboling'])
```

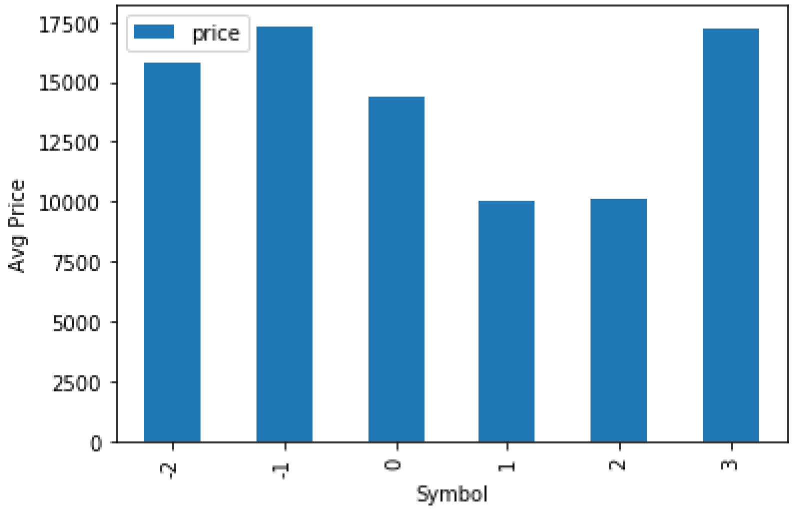
Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x294c162e1c0>
```



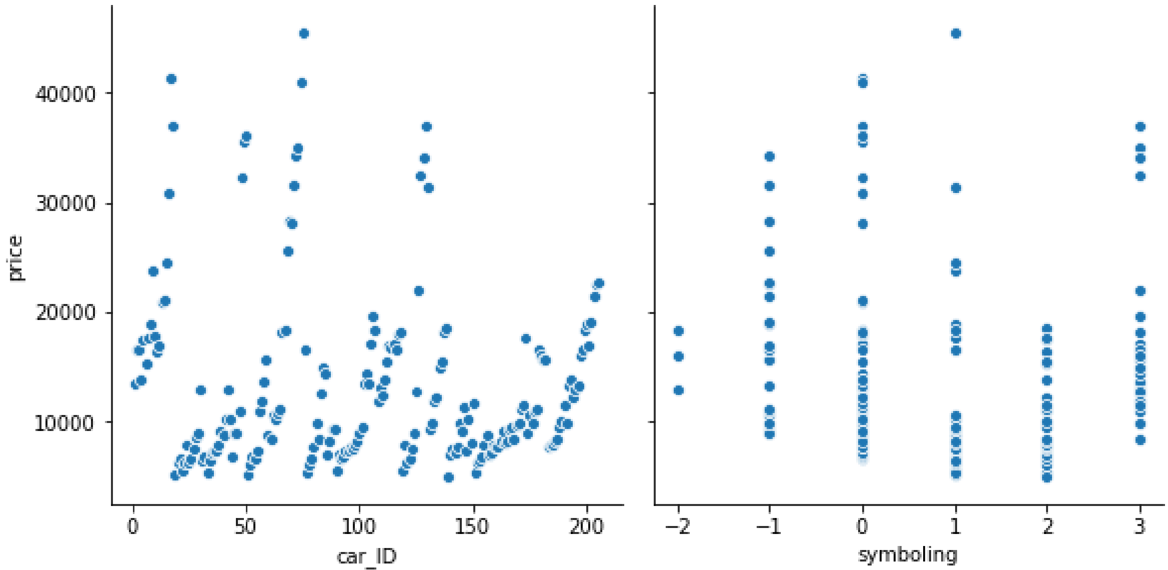
In [100]:

```
cars[['symboling','price']].groupby("symboling").mean().plot(kind='bar')
plt.xlabel("Symbol")
plt.ylabel("Avg Price")
plt.show()
```



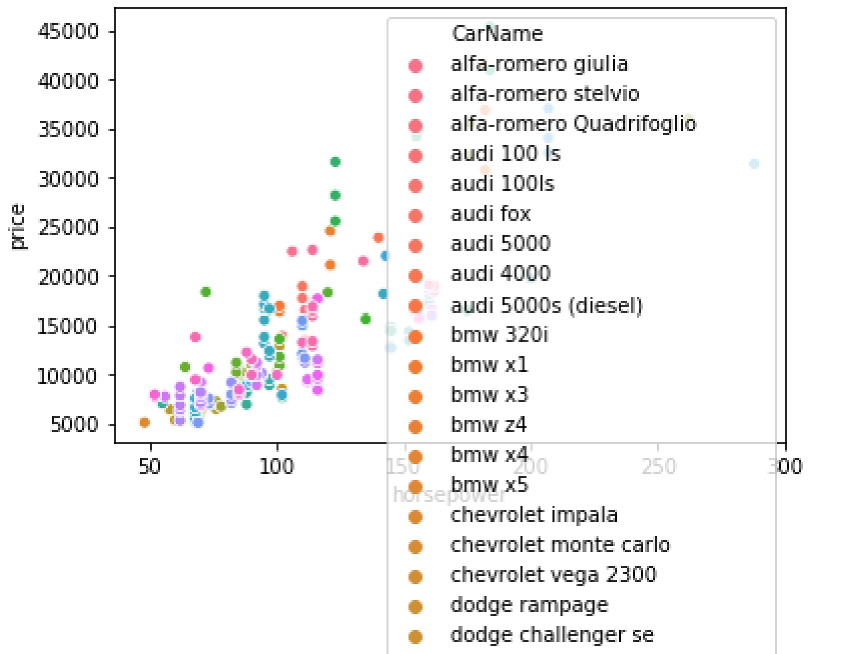

In [101]:

```
sn.pairplot(cars, x_vars=['car_ID', 'symboling'], y_vars='price',size=4, aspect=1, kind='sc
plt.show()
```



In [102]:

```
plt = sn.scatterplot(x = 'horsepower', y = 'price', hue = 'CarName', data = cars)
```

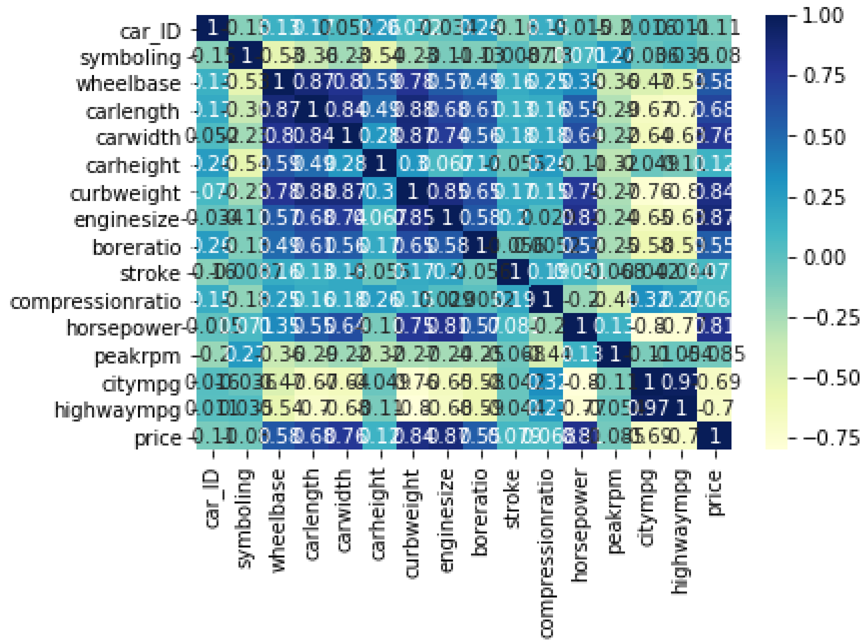



In [103]:

```
sn.heatmap(cars.corr(), cmap="YlGnBu", annot = True)
```

Out[103]:

<matplotlib.axes._subplots.AxesSubplot at 0x294bfe81880>



In [104]:

```
cars.CarName.values[0:15]
```

Out[104]:

```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 100ls', 'audi 5000', 'audi 4000',
       'audi 5000s (diesel)', 'bmw 320i', 'bmw 320i', 'bmw x1', 'bmw x3',
       'bmw z4'], dtype=object)
```

# Data preparation

In [105]:

```
cars.CarName.values[0:5]
```

Out[105]:

```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls'],
      dtype=object)
```

In [106]:

```
CompanyName = cars['CarName'].apply(lambda x : x.split(' ')[0])
cars.insert(3,"CompanyName",CompanyName)
cars.drop(['CarName'],axis=1,inplace=True)
cars.head()
```

Out[106]:

| | car_ID | symboling | CompanyName | fueltype | aspiration | doornumber | carbody | drivewheel |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 3 | alfa-romero | gas | std | two | convertible | rwd |
| **1** | 2 | 3 | alfa-romero | gas | std | two | convertible | rwd |
| **2** | 3 | 1 | alfa-romero | gas | std | two | hatchback | rwd |
| **3** | 4 | 2 | audi | gas | std | four | sedan | fwd |
| **4** | 5 | 2 | audi | gas | std | four | sedan | 4wd |

5 rows × 26 columns

In [107]:

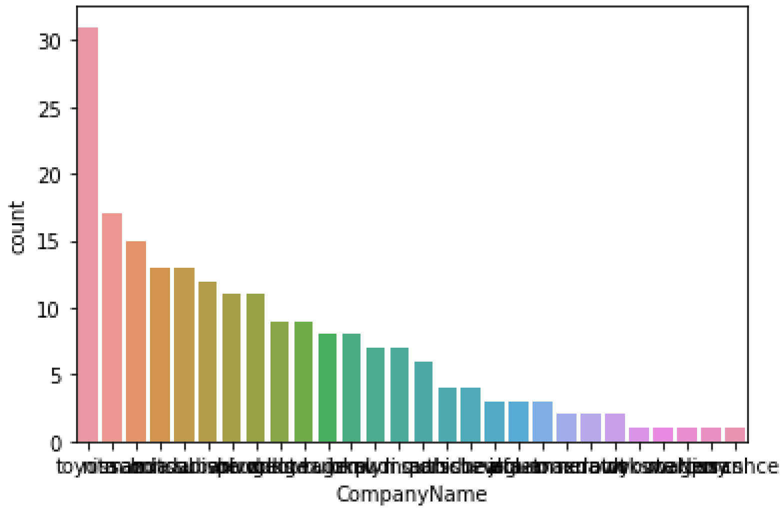
```
cars.CompanyName.unique()
```

Out[107]:

```
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 [108]:

```
plt.plot(figsize=(10,8))
sn.countplot(cars['CompanyName'], order=pd.value_counts(cars['CompanyName']).index,)
plt.xlabel = 'company name'
plt.ylabel= 'Count of Cars'
```




In [109]:

```
cars.CompanyName.describe()
```

Out[109]:

```
count        205
unique        28
top       toyota
freq          31
Name: CompanyName, dtype: object
```

In [110]:

```
cars.price.describe()
```

Out[110]:

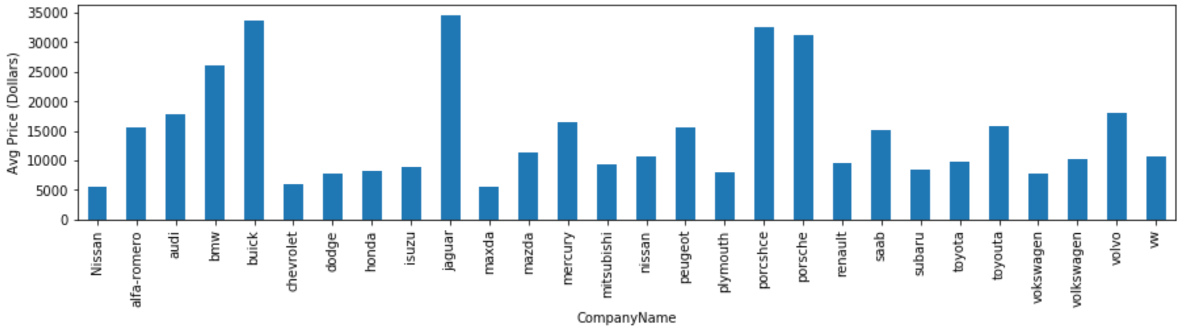
```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
25%       7788.000000
50%      10295.000000
75%      16503.000000
max      45400.000000
Name: price, dtype: float64
```

In [111]:

```
cars_comp_avg_price = cars[['CompanyName','price']].groupby("CompanyName", as_index = False
plt = cars_comp_avg_price.plot(x = 'CompanyName', kind='bar',legend = False, sort_columns =
plt.set_xlabel("CompanyName")
plt.set_ylabel("Avg Price (Dollars)")
```

Out[111]:

```
Text(0, 0.5, 'Avg Price (Dollars)')
```



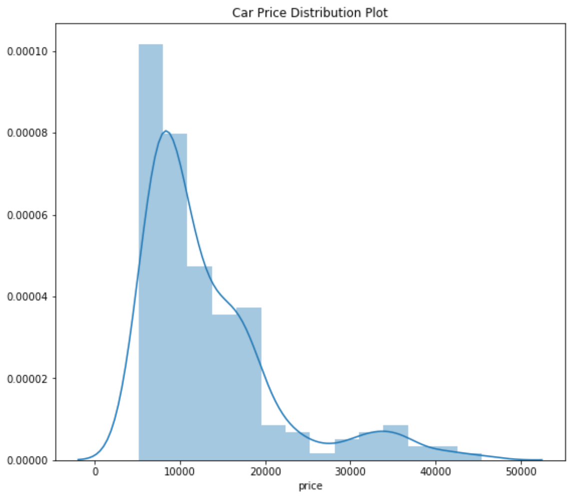

In [115]:

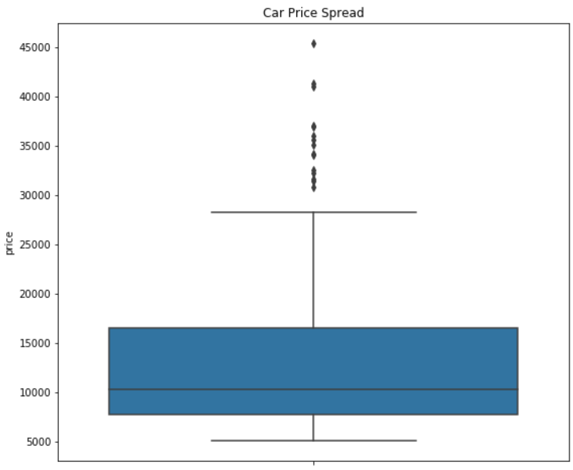
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Car Price Distribution Plot')
sn.distplot(cars.price)

plt.subplot(1,2,2)
plt.title('Car Price Spread')
sn.boxplot(y=cars.price)

plt.show()
```



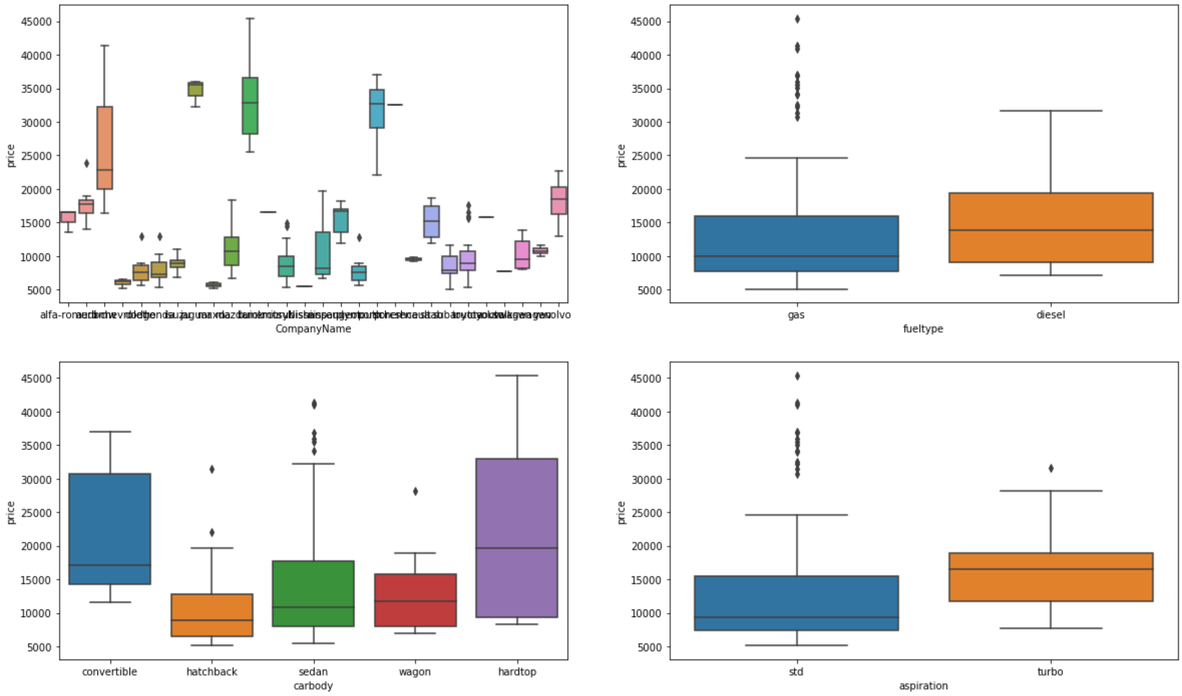



In [116]:

```
plt.figure(figsize=(20, 12))
plt.subplot(2,2,1)
sn.boxplot(x = 'CompanyName', y = 'price', data = cars)
plt.subplot(2,2,2)
sn.boxplot(x = 'fueltype', y = 'price', data = cars)
plt.subplot(2,2,3)
sn.boxplot(x = 'carbody', y = 'price', data = cars)
plt.subplot(2,2,4)
sn.boxplot(x = 'aspiration', y = 'price', data = cars)
```

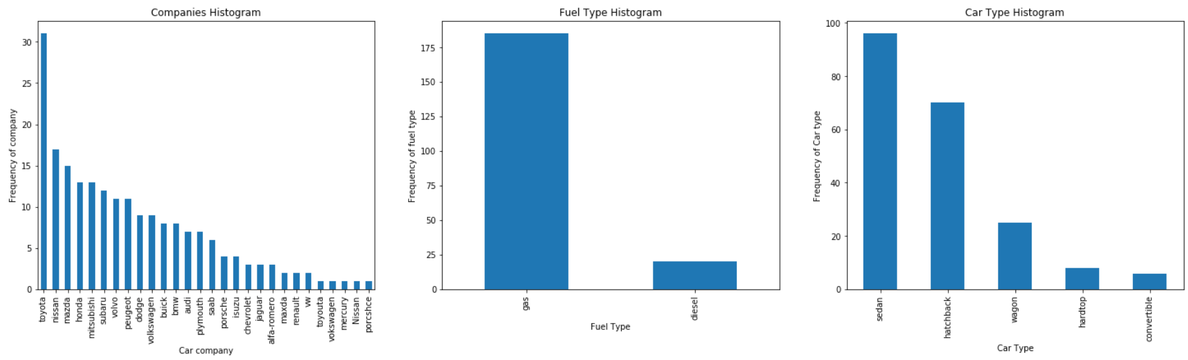
Out[116]:

<matplotlib.axes._subplots.AxesSubplot at 0x294b836fb20>



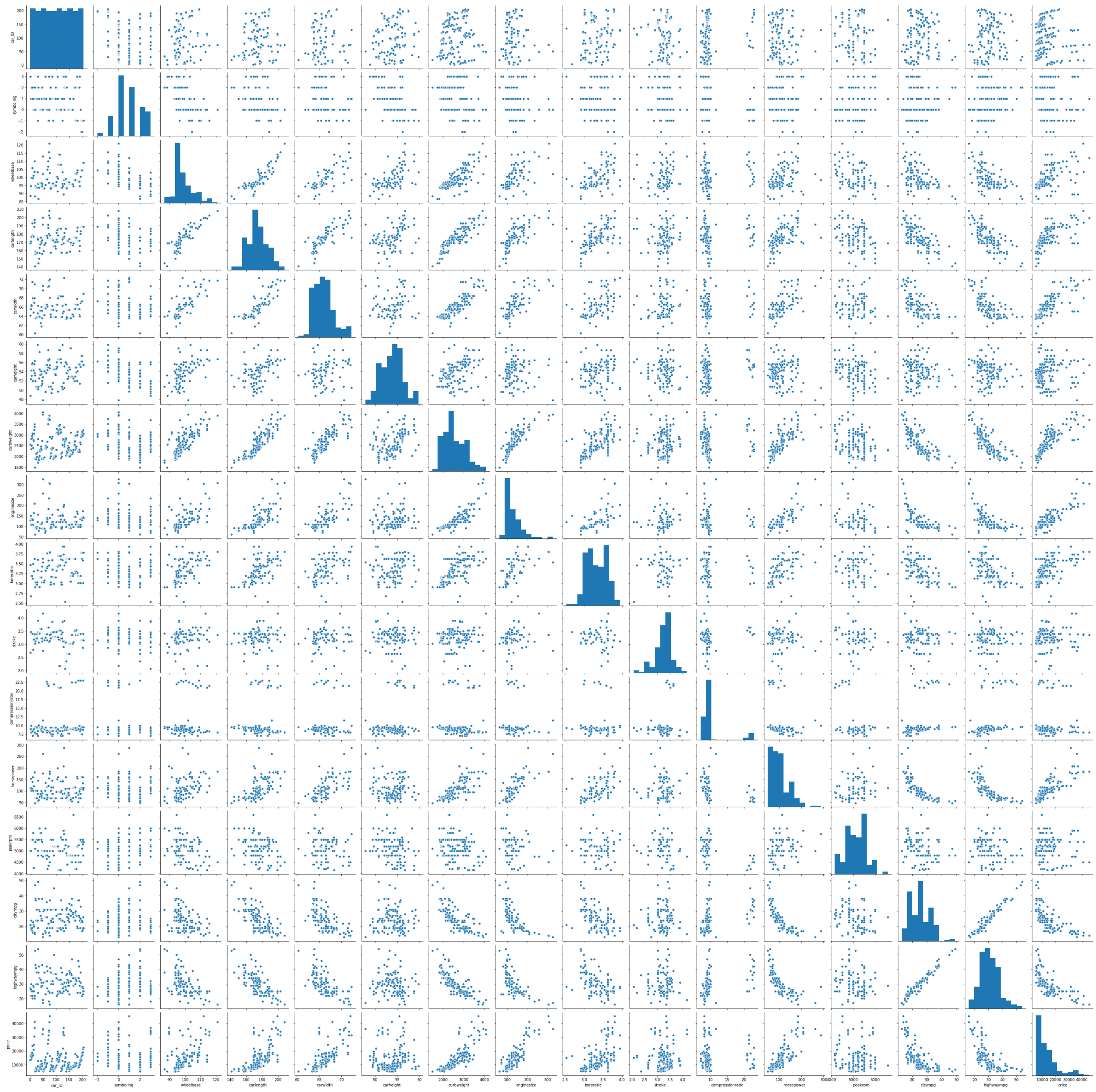
In [117]:

```
plt.figure(figsize=(25, 6))
plt.subplot(1,3,1)
plt1 = cars.CompanyName.value_counts().plot(kind='bar')
plt.title('Companies Histogram')
plt1.set(xlabel = 'Car company', ylabel='Frequency of company')
plt.subplot(1,3,2)
plt1 = cars.fueltype.value_counts().plot(kind='bar')
plt.title('Fuel Type Histogram')
plt1.set(xlabel = 'Fuel Type', ylabel='Frequency of fuel type')
plt.subplot(1,3,3)
plt1 = cars.carbody.value_counts().plot(kind='bar')
plt.title('Car Type Histogram')
plt1.set(xlabel = 'Car Type', ylabel='Frequency of Car type')
plt.show()
```



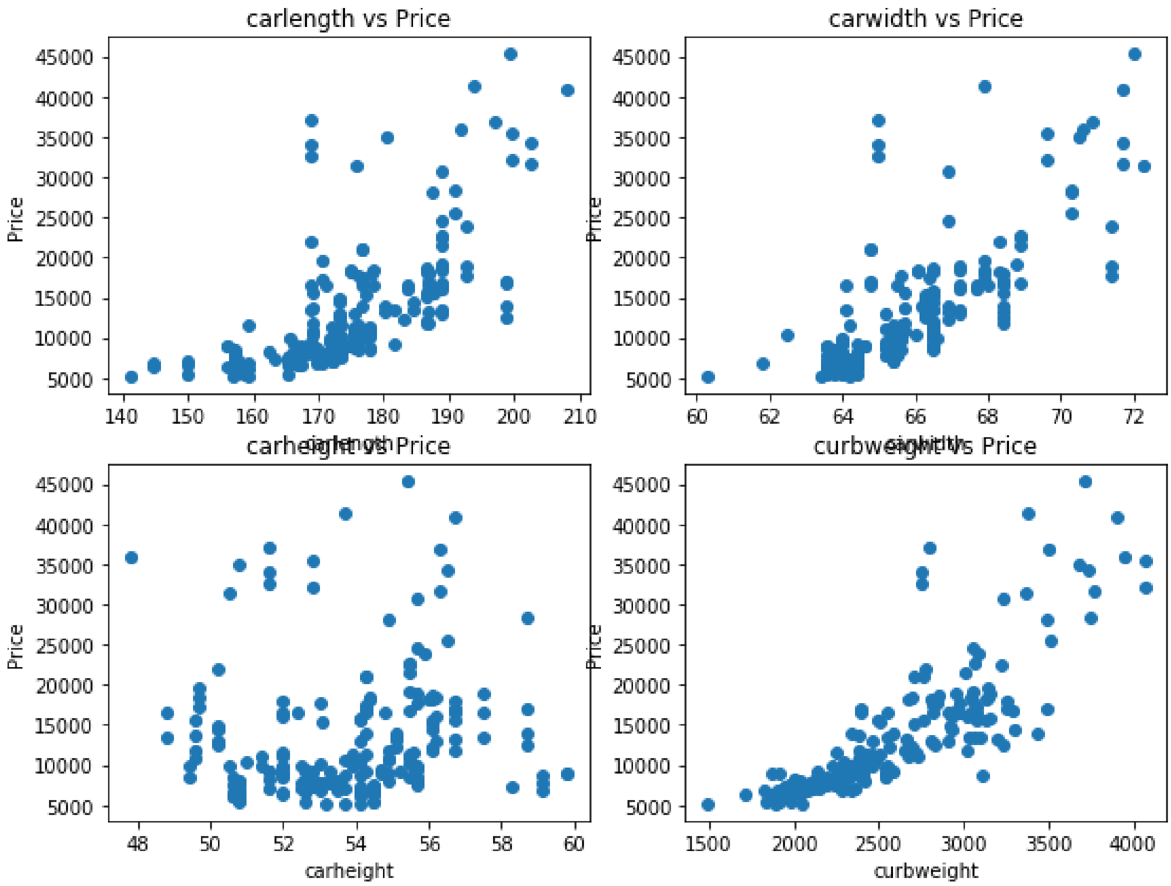
In [35]:

```
sn.pairplot(cars)
plt.show()
```



In [118]:

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



In [119]:

```
cars_lr = cars[['price', 'fueltype', 'aspiration','carbody', 'drivewheel','wheelbase','curb
                'cylindernumber', 'enginesize', 'boreratio','horsepower',  'carlength','car
cars_lr.head()
```

Out[119]:

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

# Dummy variables

In [120]:

```
status = pd.get_dummies(cars['cylindernumber'])
status.head()
```

Out[120]:

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

In [121]:

```
status = pd.get_dummies(cars['cylindernumber'], drop_first = True)
cars= pd.concat([cars, status], axis = 1)
cars.head()
```

Out[121]:

| | car_ID | symboling | CompanyName | fueltype | aspiration | doornumber | carbody | drivewheel |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 3 | alfa-romero | gas | std | two | convertible | rwd |
| **1** | 2 | 3 | alfa-romero | gas | std | two | convertible | rwd |
| **2** | 3 | 1 | alfa-romero | gas | std | two | hatchback | rwd |
| **3** | 4 | 2 | audi | gas | std | four | sedan | fwd |
| **4** | 5 | 2 | audi | gas | std | four | sedan | 4wd |

5 rows × 32 columns

# Train_Test spliting and feature selection

In [149]:

```
from sklearn.model_selection import train_test_split
np.random.seed(0)
df_train, df_test = train_test_split(cars_lr, train_size = 0.7, test_size = 0.3, random_sta
```

In [150]:

```
df_train.head()
```

Out[150]:

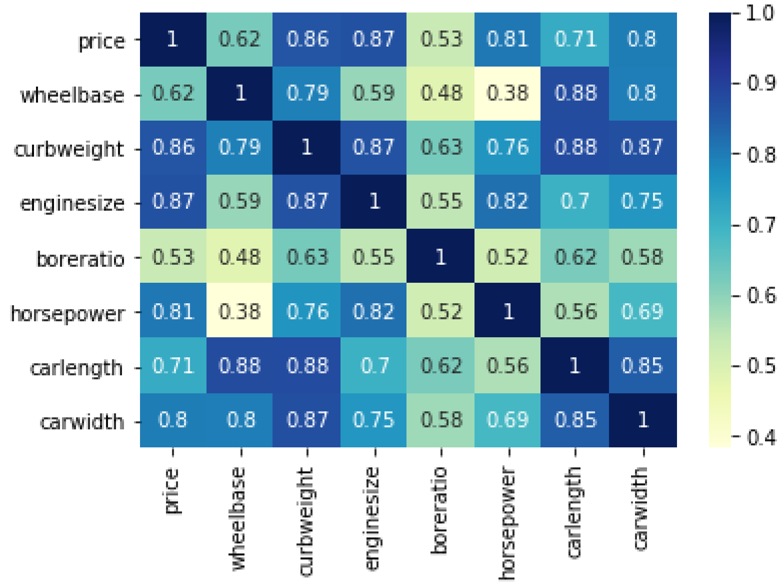
| | price | fueltype | aspiration | carbody | drivewheel | wheelbase | curbweight | enginetype |
|---|---|---|---|---|---|---|---|---|
| **122** | 7609.0 | gas | std | sedan | fwd | 93.7 | 2191 | ohc |
| **125** | 22018.0 | gas | std | hatchback | rwd | 94.5 | 2778 | ohc |
| **166** | 9538.0 | gas | std | hatchback | rwd | 94.5 | 2300 | dohc |
| **1** | 16500.0 | gas | std | convertible | rwd | 88.6 | 2548 | dohc |
| **199** | 18950.0 | gas | turbo | wagon | rwd | 104.3 | 3157 | ohc |

In [151]:

```
plt.figure.Figsize =(30, 10)
sn.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
```

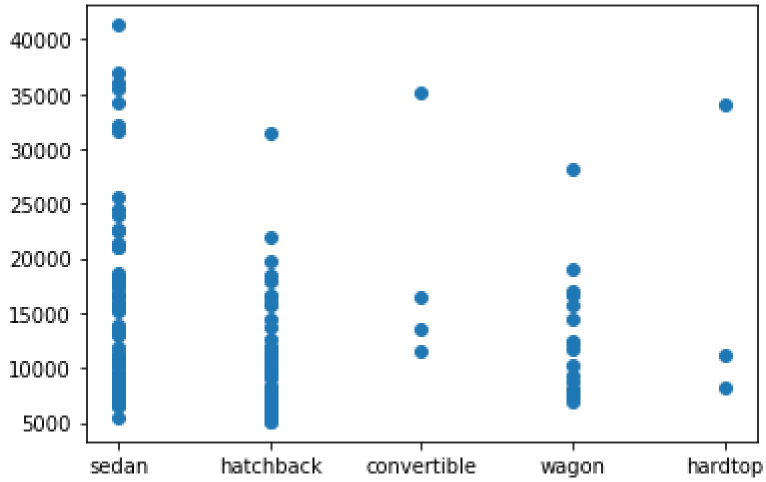
Out[151]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x294c0fb6550>
```



In [152]:

```
plt.figure.figsize=[6,6]
plt.scatter(df_train.carbody, df_train.price)
plt.show()
```




In [153]:

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

# Model building

In [154]:

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

In [155]:

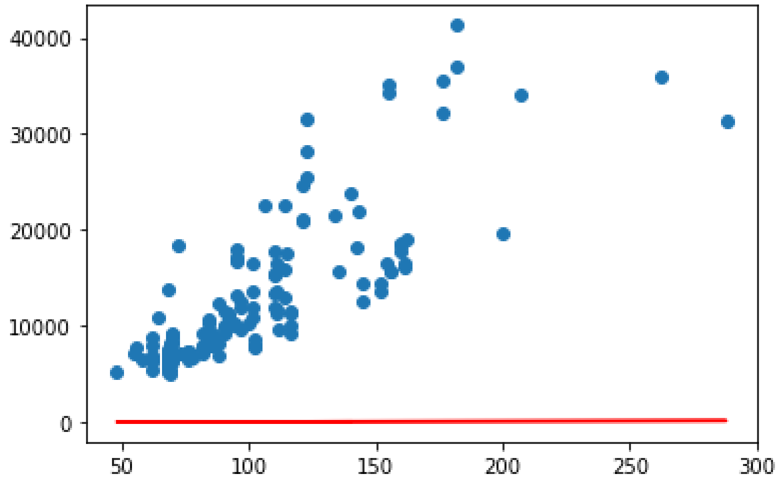
```
lr.params
```

Out[155]:

```
const        -3192.650565
horsepower     158.445735
dtype: float64
```

In [156]:

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



In [157]:

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

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                  price   R-squared:                       0.6
50
Model:                            OLS   Adj. R-squared:                  0.6
47
Method:                 Least Squares   F-statistic:                      26
1.8
Date:                Sat, 25 Apr 2020   Prob (F-statistic):           6.04e-
34
Time:                        14:29:50   Log-Likelihood:                 -140
9.0
No. Observations:                 143   AIC:                             282
2.
Df Residuals:                     141   BIC:                             282
8.
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const      -3192.6506   1076.508     -2.966      0.004   -5320.833   -1064.4
69
horsepower   158.4457      9.793     16.180      0.000     139.086     177.8
06
==============================================================================
==
Omnibus:                       33.630   Durbin-Watson:                   1.8
32
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.5
78
Skew:                           1.166   Prob(JB):                     2.32e-
12
Kurtosis:                       4.886   Cond. No.                         30
5.
==============================================================================
==

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

In the above R-squared value obtained is 0.659.Since we have so many variables, we can clearly do better than this.

In [158]:

```
X_train_lm = X_train[['horsepower', 'boreratio']]
```

In [159]:

```
import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train_lm)
lr = sm.OLS(y_train, X_train_lm).fit()
lr.params
```

Out[159]:

```
const        -17195.969523
horsepower      142.324775
boreratio      4733.779792
dtype: float64
```

In [160]:

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

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                  price   R-squared:                       0.6
68
Model:                            OLS   Adj. R-squared:                  0.6
64
Method:                 Least Squares   F-statistic:                     14
1.0
Date:                Sat, 25 Apr 2020   Prob (F-statistic):           2.86e-
34
Time:                        14:29:53   Log-Likelihood:                -140
5.2
No. Observations:                 143   AIC:                             281
6.
Df Residuals:                     140   BIC:                             282
5.
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const       -1.72e+04   5145.399     -3.342      0.001   -2.74e+04   -7023.2
40
horsepower   142.3248     11.187     12.722      0.000     120.207     164.4
43
boreratio   4733.7798   1702.665      2.780      0.006    1367.520    8100.0
40
==============================================================================
==
Omnibus:                       33.546   Durbin-Watson:                   1.7
73
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.8
88
Skew:                           1.156   Prob(JB):                     1.99e-
12
Kurtosis:                       4.923   Cond. No.                     1.57e+
03
==============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.57e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [161]:

```
X_train_lm = X_train[['horsepower', 'boreratio','wheelbase']]
```

In [162]:

```
import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train_lm)
lr = sm.OLS(y_train, X_train_lm).fit()
lr.params
```

Out[162]:

```
const        -48672.038019
horsepower      129.410936
boreratio       528.484193
wheelbase       474.092601
dtype: float64
```

In [163]:

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

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                  price   R-squared:                       0.7
66
Model:                            OLS   Adj. R-squared:                  0.7
61
Method:                 Least Squares   F-statistic:                     15
1.4
Date:                Sat, 25 Apr 2020   Prob (F-statistic):           1.33e-
43
Time:                        14:29:56   Log-Likelihood:                -138
0.3
No. Observations:                 143   AIC:                             276
9.
Df Residuals:                     139   BIC:                             278
1.
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const      -4.867e+04   5998.451     -8.114      0.000   -6.05e+04   -3.68e+
04
horsepower   129.4109      9.588     13.497      0.000     110.454     148.3
68
boreratio    528.4842   1539.016      0.343      0.732   -2514.424    3571.3
92
wheelbase    474.0926     62.368      7.602      0.000     350.780     597.4
05
==============================================================================
==
Omnibus:                       42.724   Durbin-Watson:                   1.8
44
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              104.4
54
Skew:                           1.214   Prob(JB):                     2.08e-
23
Kurtosis:                       6.411   Cond. No.                     2.75e+
03
==============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 2.75e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

# Checking VIF

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

$VIFi$=1/1−$Ri$2

In [164]:

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

In [166]:

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

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

In [168]:

```
X_train_new = build_model(X_train_lm,y_train)
```

```
                            OLS Regression Results                            
============================================================================
==
Dep. Variable:                  price   R-squared:                       0.7
66
Model:                            OLS   Adj. R-squared:                  0.7
61
Method:                 Least Squares   F-statistic:                      15
1.4
Date:                Sat, 25 Apr 2020   Prob (F-statistic):           1.33e-
43
Time:                        14:35:59   Log-Likelihood:                -138
0.3
No. Observations:                 143   AIC:                             276
9.
Df Residuals:                     139   BIC:                             278
1.
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------
--
const      -4.867e+04   5998.451     -8.114      0.000   -6.05e+04   -3.68e+
04
horsepower   129.4109      9.588     13.497      0.000     110.454     148.3
68
boreratio    528.4842   1539.016      0.343      0.732   -2514.424    3571.3
92
wheelbase    474.0926     62.368      7.602      0.000     350.780     597.4
05
============================================================================
==
Omnibus:                       42.724   Durbin-Watson:                   1.8
44
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              104.4
54
Skew:                           1.214   Prob(JB):                     2.08e-
23
Kurtosis:                       6.411   Cond. No.                     2.75e+
03
============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 2.75e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [169]:

```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
============================================================================
==
Dep. Variable:                  price   R-squared:                       0.7
66
Model:                            OLS   Adj. R-squared:                  0.7
61
Method:                 Least Squares   F-statistic:                     15
1.4
Date:                Sat, 25 Apr 2020   Prob (F-statistic):           1.33e-
43
Time:                        14:36:35   Log-Likelihood:                -138
0.3
No. Observations:                 143   AIC:                             276
9.
Df Residuals:                     139   BIC:                             278
1.
Df Model:                           3
Covariance Type:            nonrobust
============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
----------------------------------------------------------------------------
--
const      -4.867e+04   5998.451     -8.114      0.000   -6.05e+04   -3.68e+
04
horsepower   129.4109      9.588     13.497      0.000     110.454     148.3
68
boreratio    528.4842   1539.016      0.343      0.732   -2514.424    3571.3
92
wheelbase    474.0926     62.368      7.602      0.000     350.780     597.4
05
============================================================================
==
Omnibus:                       42.724   Durbin-Watson:                   1.8
44
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              104.4
54
Skew:                           1.214   Prob(JB):                     2.08e-
23
Kurtosis:                       6.411   Cond. No.                     2.75e+
03
============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 2.75e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [171]:

```
checkVIF(X_train_new)
```

Out[171]:

| | Features | VIF |
|---|---|---|
| 0 | const | 352.64 |
| 2 | boreratio | 1.57 |
| 1 | horsepower | 1.41 |
| 3 | wheelbase | 1.35 |

In [172]:

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

In [173]:

```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                  price   R-squared:                       0.6
68
Model:                            OLS   Adj. R-squared:                  0.6
64
Method:                 Least Squares   F-statistic:                     14
1.0
Date:                Sat, 25 Apr 2020   Prob (F-statistic):           2.86e-
34
Time:                        14:38:54   Log-Likelihood:                -140
5.2
No. Observations:                 143   AIC:                             281
6.
Df Residuals:                     140   BIC:                             282
5.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const      -1.72e+04   5145.399     -3.342      0.001   -2.74e+04   -7023.2
40
horsepower   142.3248     11.187     12.722      0.000     120.207     164.4
43
boreratio   4733.7798   1702.665      2.780      0.006    1367.520    8100.0
40
==============================================================================
==
Omnibus:                       33.546   Durbin-Watson:                   1.7
73
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.8
88
Skew:                           1.156   Prob(JB):                     1.99e-
12
Kurtosis:                       4.923   Cond. No.                     1.57e+
03
==============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.57e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [174]:

```
checkVIF(X_train_new)
```

Out[174]:

| | Features | VIF |
|---|---|---|
| 0 | const | 184.60 |
| 1 | horsepower | 1.37 |
| 2 | boreratio | 1.37 |

# Residual analysis of a model

In [175]:

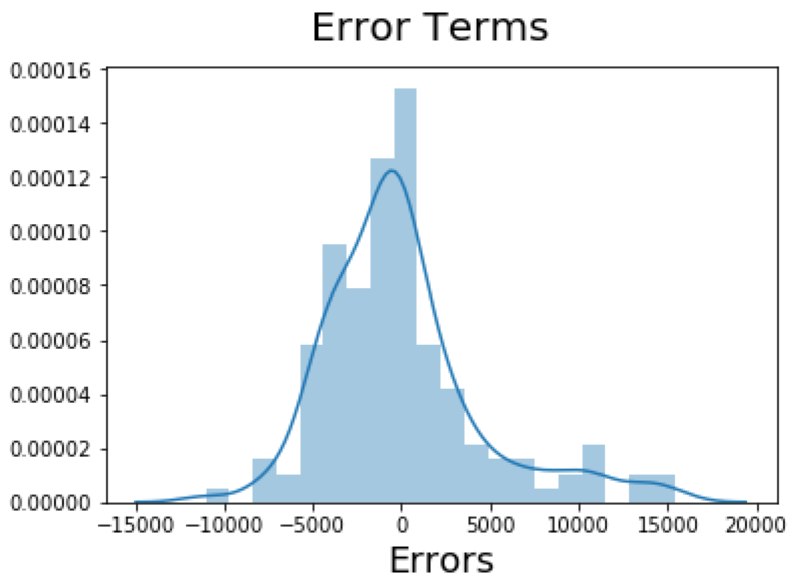
```
lm = sm.OLS(y_train,X_train_new).fit()
y_train_price = lm.predict(X_train_new)
```

In [177]:

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

Out[177]:

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




# Model evaluation

In [185]:

```
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepower','fueleconomy
df_test[num_vars] = scaler.transform(df_test[num_vars])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-185-19e1deb196ac> in <module>
      1 num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'h
orsepower','fueleconomy','carlength','carwidth','price']
----> 2 df_test[num_vars] = scaler.transform(df_test[num_vars])

NameError: name 'scaler' is not defined
```

In [186]:

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

In [187]:

```
X_train_new = X_train_new.drop('const',axis=1)
X_test_new = X_test[X_train_new.columns]
X_test_new = sm.add_constant(X_test_new)
```

In [188]:

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

In [189]:

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

Out[189]:

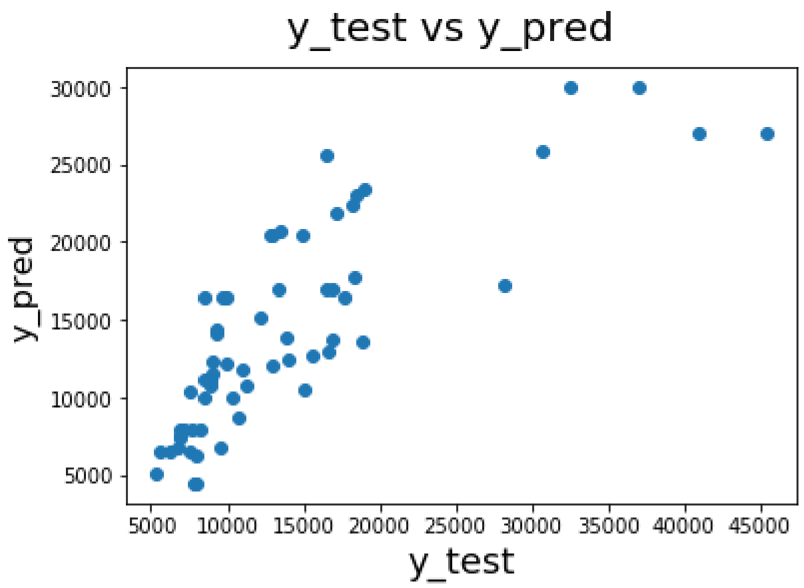
```
0.6544700407307734
```

In [190]:

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

Out[190]:

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



In [191]:

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

```
                            OLS Regression Results                            
==============================================================================
==
Dep. Variable:                  price   R-squared:                       0.6
68
Model:                            OLS   Adj. R-squared:                  0.6
64
Method:                 Least Squares   F-statistic:                     14
1.0
Date:                Sat, 25 Apr 2020   Prob (F-statistic):           2.86e-
34
Time:                        15:02:11   Log-Likelihood:                -140
5.2
No. Observations:                 143   AIC:                             281
6.
Df Residuals:                     140   BIC:                             282
5.
Df Model:                           2                                         
Covariance Type:            nonrobust                                         
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
const        -1.72e+04   5145.399     -3.342      0.001   -2.74e+04   -7023.2
40
horsepower    142.3248     11.187     12.722      0.000     120.207     164.4
43
boreratio    4733.7798   1702.665      2.780      0.006    1367.520    8100.0
40
==============================================================================
==
Omnibus:                       33.546   Durbin-Watson:                   1.7
73
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.8
88
Skew:                           1.156   Prob(JB):                     1.99e-
12
Kurtosis:                       4.923   Cond. No.                     1.57e+
03
==============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The condition number is large, 1.57e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

In [ ]: