In [1]:
```
import warnings
warnings.filterwarnings('ignore')

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

In [2]:
```
auto=pd.read_csv('C:/Users/yoges/Desktop/AutoData (1).csv')
auto.head()
```

Out[2]:

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

5 rows × 25 columns

In [3]:
```
auto.shape
```

Out[3]:
```
(205, 25)
```

In [4]:
```
auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   symboling         205 non-null    int64  
 1   make              205 non-null    object 
 2   fueltype          205 non-null    object 
 3   aspiration        205 non-null    object 
 4   doornumber        205 non-null    object 
 5   carbody           205 non-null    object 
 6   drivewheel        205 non-null    object 
 7   enginelocation    205 non-null    object 
 8   wheelbase         205 non-null    float64
 9   carlength         205 non-null    float64
 10  carwidth          205 non-null    float64
 11  carheight         205 non-null    float64
 12  curbweight        205 non-null    int64  
 13  enginetype        205 non-null    object 
 14  cylindernumber    205 non-null    object 
 15  enginesize        205 non-null    int64  
 16  fuelsystem        205 non-null    object 
 17  boreratio         205 non-null    float64
 18  stroke            205 non-null    float64
 19  compressionratio  205 non-null    float64
 20  horsepower        205 non-null    int64  
 21  peakrpm           205 non-null    int64  
 22  citympg           205 non-null    int64  
 23  highwaympg        205 non-null    int64  
 24  price             205 non-null    float64
dtypes: float64(8), int64(7), object(10)
memory usage: 40.2+ KB
```

In [5]:
```
auto.isnull().sum()
```

Out[5]:
```
symboling           0
make                0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

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

Out[6]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	bor
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.3
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.2
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.5
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.1
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.3
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.5
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.9

In [7]:
```
auto.describe(include=object)
```

Out[7]:

	make	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype
count	205	205	205	205	205	205	205	205
unique	147	2	2	2	5	3	2	7
top	toyota corona	gas	std	four	sedan	fwd	front	ohc
freq	6	185	168	115	96	120	202	148

Data Cleaning and Preparation

In [8]:
```
Company = auto['make'].apply(lambda x : x.split(' ')[0])
auto.insert(3,"Company",Company)
auto.drop(['make'],axis=1,inplace=True)
auto.head()
```

Out[8]:

	symboling	fueltype	Company	aspiration	doornumber	carbody	drivewheel	enginelocation
0	3	gas	alfa-romero	std	two	convertible	rwd	front
1	3	gas	alfa-romero	std	two	convertible	rwd	front
2	1	gas	alfa-romero	std	two	hatchback	rwd	front
3	2	gas	audi	std	four	sedan	fwd	front
4	2	gas	audi	std	four	sedan	4wd	front

5 rows × 25 columns

In [9]:
```
auto.Company.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)
```

There seems to be some spelling error in the CompanyName column.

maxda = mazda Nissan = nissan porsche = porcshce toyota = toyouta vokswagen = volkswagen = vw so we will fix this

In [10]:
```
auto.Company = auto.Company.str.lower()

def replace_name(a,b):
    auto.Company.replace(a,b,inplace=True)

replace_name('maxda','mazda')
replace_name('porcshce','porsche')
replace_name('toyouta','toyota')
replace_name('vokswagen','volkswagen')
replace_name('vw','volkswagen')

auto.Company.unique()
```

Out[10]:
```
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 [11]:
```
#Checking for duplicates
auto.loc[auto.duplicated()]
```

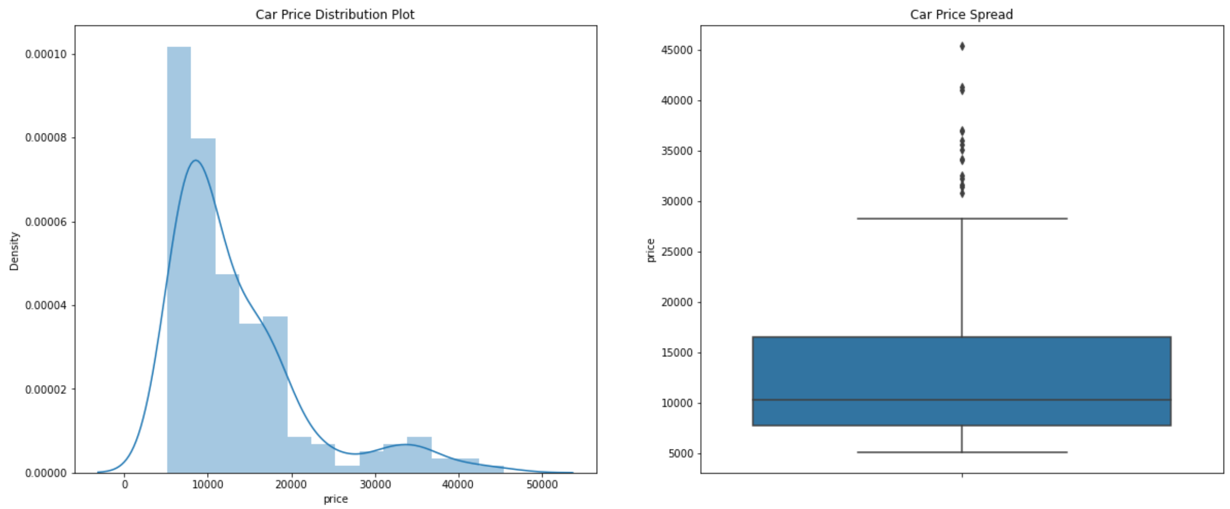
Out[11]:

symboling	fueltype	Company	aspiration	doornumber	carbody	drivewheel	enginelocation	wh

0 rows × 25 columns

Data Visulaization:

In [12]:
```
plt.figure(figsize=(20,8))

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

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

plt.show()
```

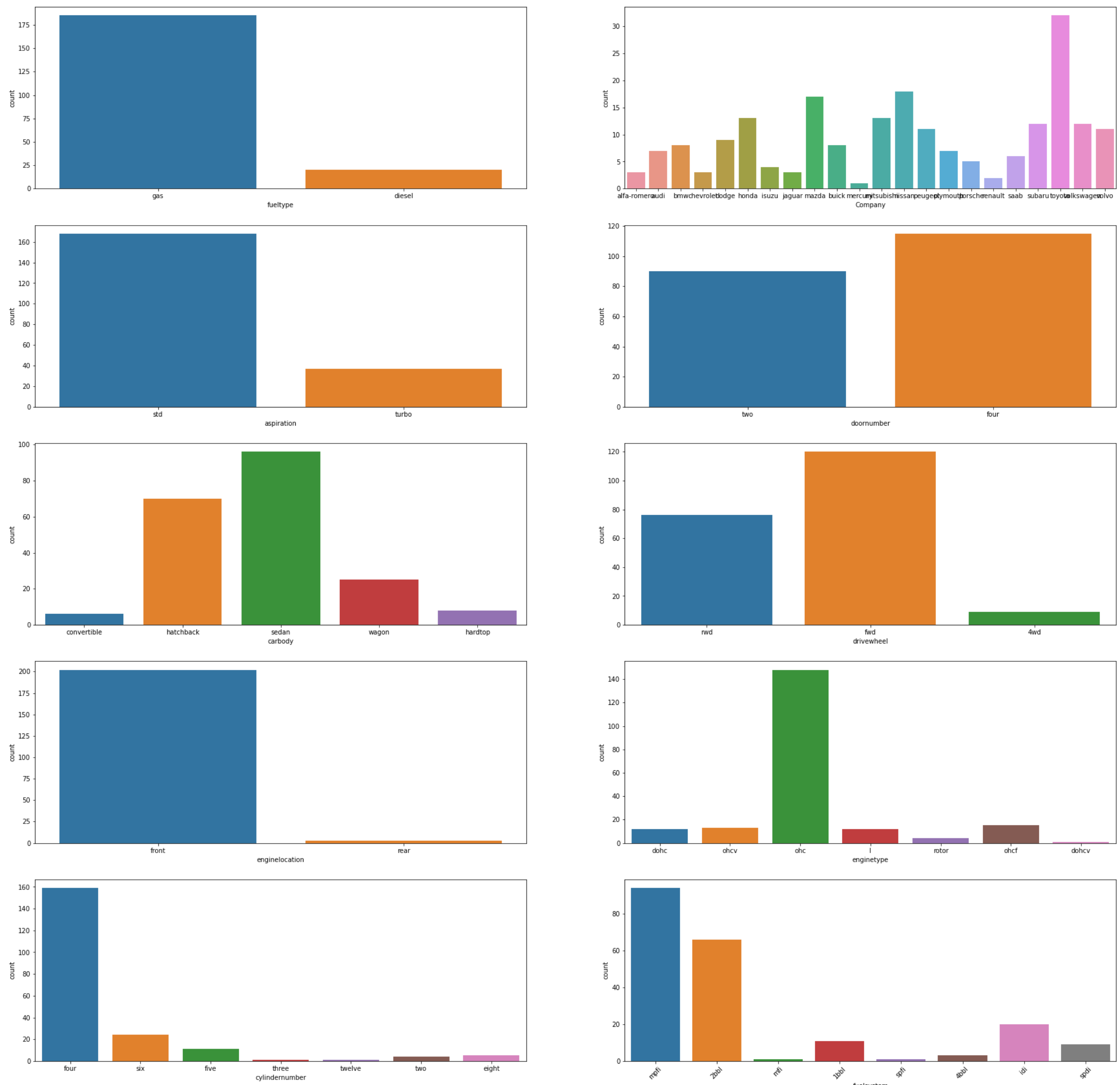


Inference : The plot seemed to be right-skewed, meaning that the most prices in the dataset are low(Below 15,000). There is a significant difference between the mean and the median of the price distribution. The data points are far spread out from the mean, which indicates a high variance in the car prices.(85% of the prices are below 18,500, whereas the remaining 15% are between 18,500 and 45,400.)

Visualising Categorical Data

fueltype Company aspiration doornumber carbody drivewheel enginelocation enginetype cylindernumber fuelsystem

In [13]:
```
obj_col = []
num_col = []
for col in auto.columns:
    if auto[col].dtype=='O':
        obj_col.append(col)
    else:
        num_col.append(col)
fig,axs = plt.subplots(5,2,figsize=(30,30))
plt.xticks(rotation=45)

c=0
for i in range(5):
  for j in range(2):
    ax=sns.countplot(data=auto,x=obj_col[c],ax=axs[i][j])
    c+=1
```

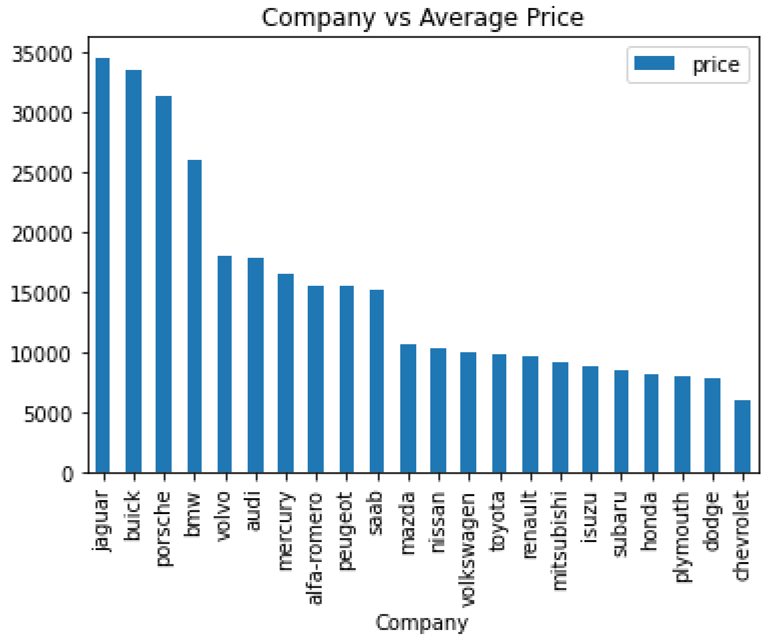


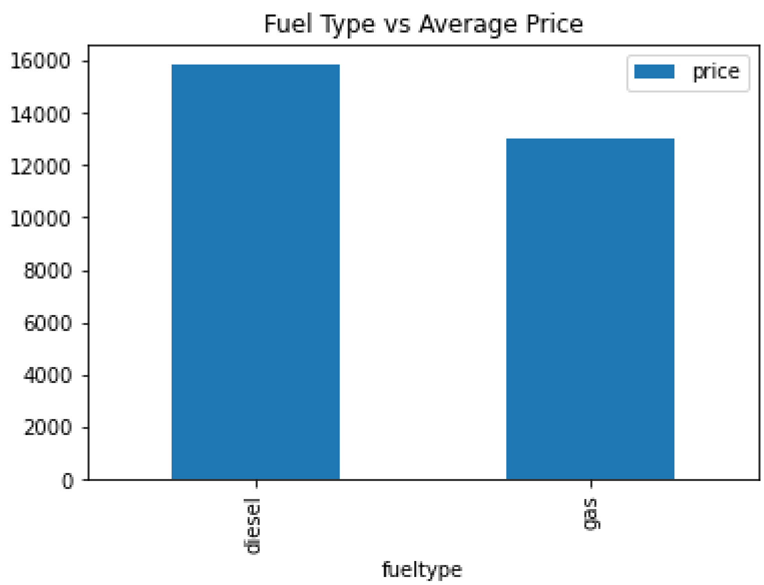
observations:

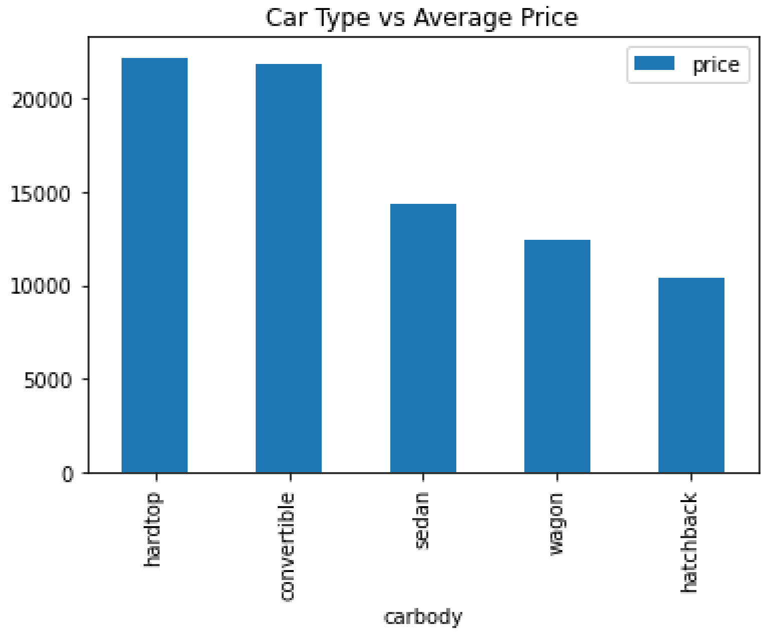
- Number of gas fueled cars are more than diesel.
- Toyota seemed to be favored car company.
- sedan is the top car type prefered.
- car with std aspiration and four door,frwrd drive wheel whit front wngine location are mostly used.
- ohc engine type with four cylinder and mpfi fuelsystem are also preferred.

In [14]:
```
plt.figure(figsize=(25, 6))

df = pd.DataFrame(auto.groupby(['Company'])['price'].mean().sort_values(ascending
df.plot.bar()
plt.title('Company vs Average Price')
plt.show()

df = pd.DataFrame(auto.groupby(['fueltype'])['price'].mean().sort_values(ascendin
df.plot.bar()
plt.title('Fuel Type vs Average Price')
plt.show()

df = pd.DataFrame(auto.groupby(['carbody'])['price'].mean().sort_values(ascending
df.plot.bar()
plt.title('Car Type vs Average Price')
plt.show()
```

```
<Figure size 1800x432 with 0 Axes>
```







Inference :

- Jaguar and Buick seem to have highest average price. diesel has higher average price than gas. hardtop and convertible have higher average price.

Visualising numerical data

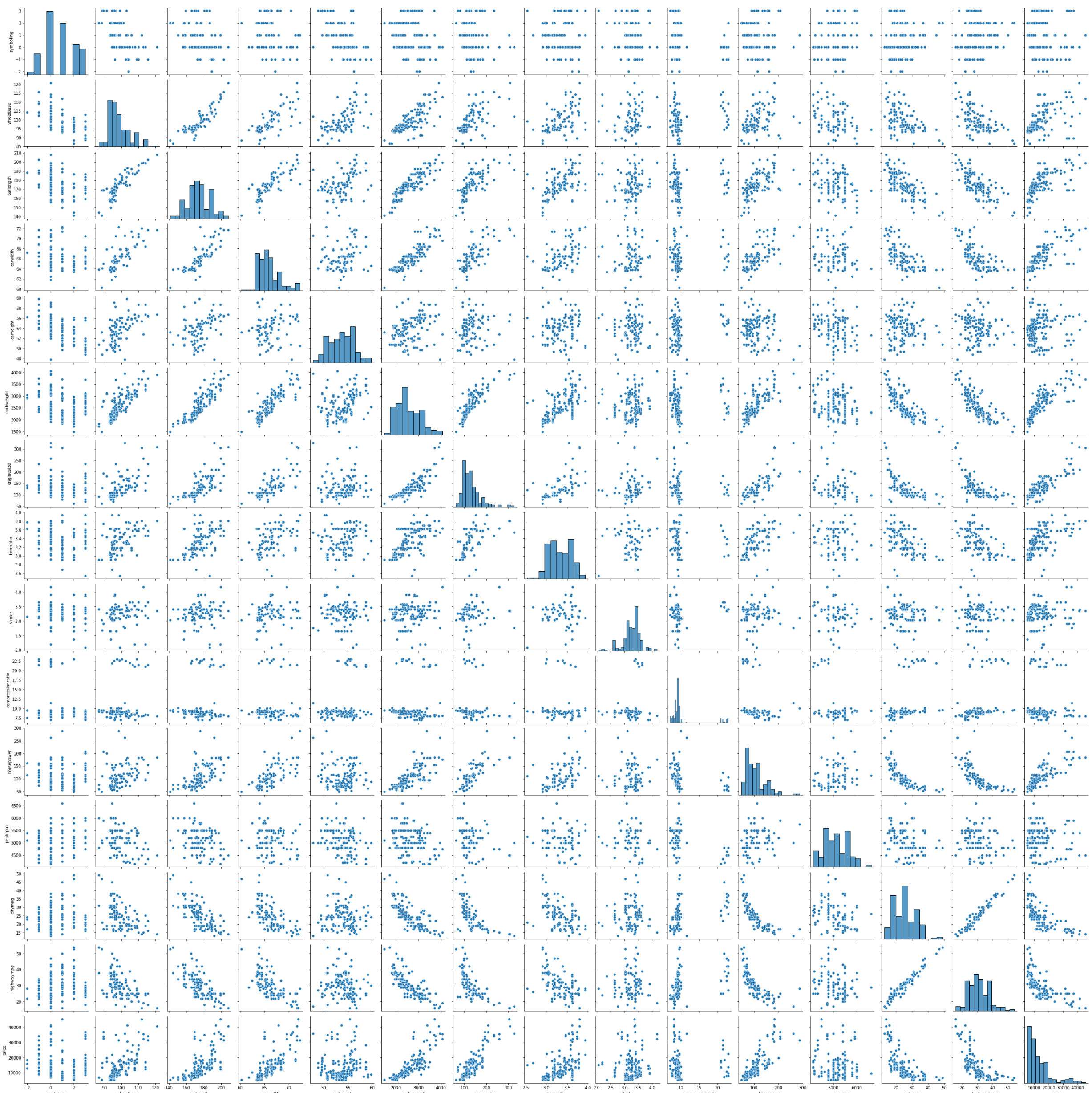
In [15]:
```
auto_numeric = auto.select_dtypes(include =['int64','float64'])
auto_numeric.head()
```

Out[15]:

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

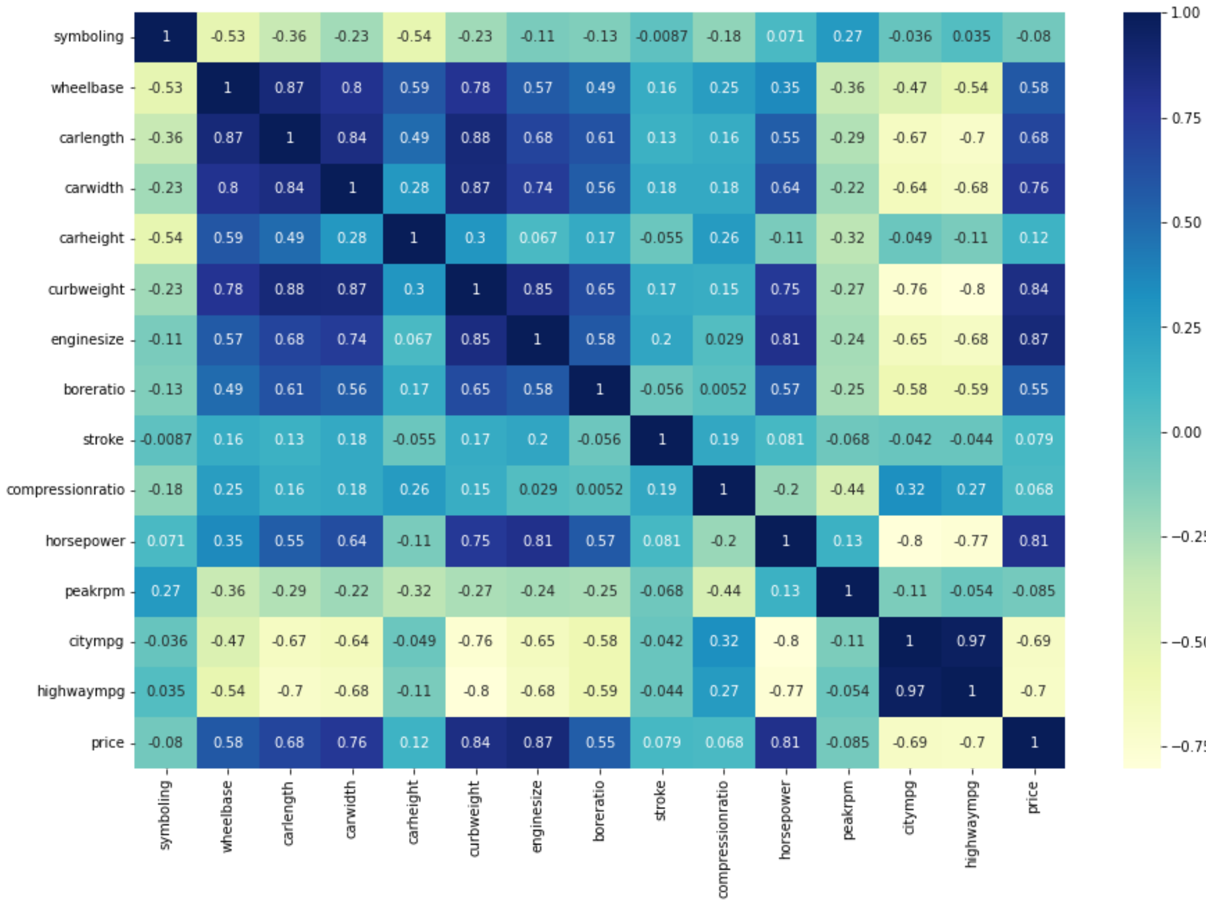
In [16]:
```
plt.figure(figsize = (30,30))
sns.pairplot(auto_numeric)
plt.show()
```

<Figure size 2160x2160 with 0 Axes>



In [17]:
```
plt.figure(figsize=(15,10))
sns.heatmap(auto.corr(),annot=True,cmap="YlGnBu")
```

Out[17]: <AxesSubplot:>



- Price is highly (positively) correlated with wheelbase, carlength, carwidth, curbweight, enginesize, horsepower.
- Price is negatively correlated to symboling, citympg and highwaympg.
- This suggest that cars having high mileage may fall in the 'economy' cars category, and are priced lower.
- There are many independent variables which are highly correlated: wheelbase, carlength, curbweight, enginesize etc.. all are positively correlated.

In [18]:
```
categorical_cols = auto.select_dtypes(include = ['object'])
categorical_cols.head()
```

Out[18]:

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

Data preparation:

In [19]:
```
#creating dummies
cars_dummies = pd.get_dummies(categorical_cols, drop_first = True)
cars_dummies.head()
```

Out[19]:

	fueltype_gas	Company_audi	Company_bmw	Company_buick	Company_chevrolet	Company_do
0	1	0	0	0	0	
1	1	0	0	0	0	
2	1	0	0	0	0	
3	1	1	0	0	0	
4	1	1	0	0	0	

5 rows × 50 columns

In [20]:
```
auto1= pd.concat([auto, cars_dummies], axis =1)
auto1 = auto1.drop(['fueltype','Company','aspiration','doornumber','carbody','dri
                    'enginelocation','enginetype','cylindernumber','fuelsystem'],
auto1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 65 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   symboling              205 non-null    int64  
 1   wheelbase              205 non-null    float64
 2   carlength              205 non-null    float64
 3   carwidth               205 non-null    float64
 4   carheight              205 non-null    float64
 5   curbweight             205 non-null    int64  
 6   enginesize             205 non-null    int64  
 7   boreratio              205 non-null    float64
 8   stroke                 205 non-null    float64
 9   compressionratio       205 non-null    float64
 10  horsepower             205 non-null    int64  
 11  peakrpm                205 non-null    int64  
 12  citympg                205 non-null    int64  
 13  highwaympg             205 non-null    int64  
 14  price                  205 non-null    float64
 15  fueltype_gas           205 non-null    uint8  
 16  Company_audi           205 non-null    uint8  
 17  Company_bmw            205 non-null    uint8  
 18  Company_buick          205 non-null    uint8  
 19  Company_chevrolet      205 non-null    uint8  
 20  Company_dodge          205 non-null    uint8  
 21  Company_honda          205 non-null    uint8  
 22  Company_isuzu          205 non-null    uint8  
 23  Company_jaguar         205 non-null    uint8  
 24  Company_mazda          205 non-null    uint8  
 25  Company_mercury        205 non-null    uint8  
 26  Company_mitsubishi     205 non-null    uint8  
 27  Company_nissan         205 non-null    uint8  
 28  Company_peugeot        205 non-null    uint8  
 29  Company_plymouth       205 non-null    uint8  
 30  Company_porsche        205 non-null    uint8  
 31  Company_renault        205 non-null    uint8  
 32  Company_saab           205 non-null    uint8  
 33  Company_subaru         205 non-null    uint8  
 34  Company_toyota         205 non-null    uint8  
 35  Company_volkswagen     205 non-null    uint8  
 36  Company_volvo          205 non-null    uint8  
 37  aspiration_turbo       205 non-null    uint8  
 38  doornumber_two         205 non-null    uint8  
 39  carbody_hardtop        205 non-null    uint8  
 40  carbody_hatchback      205 non-null    uint8  
 41  carbody_sedan          205 non-null    uint8  
 42  carbody_wagon          205 non-null    uint8  
 43  drivewheel_fwd         205 non-null    uint8  
 44  drivewheel_rwd         205 non-null    uint8  
 45  enginelocation_rear    205 non-null    uint8  
```

```
 46  enginetype_dohcv       205 non-null    uint8
 47  enginetype_l           205 non-null    uint8
 48  enginetype_ohc         205 non-null    uint8
 49  enginetype_ohcf        205 non-null    uint8
 50  enginetype_ohcv        205 non-null    uint8
 51  enginetype_rotor       205 non-null    uint8
 52  cylindernumber_five    205 non-null    uint8
 53  cylindernumber_four    205 non-null    uint8
 54  cylindernumber_six     205 non-null    uint8
 55  cylindernumber_three   205 non-null    uint8
 56  cylindernumber_twelve  205 non-null    uint8
 57  cylindernumber_two     205 non-null    uint8
 58  fuelsystem_2bbl        205 non-null    uint8
 59  fuelsystem_4bbl        205 non-null    uint8
 60  fuelsystem_idi         205 non-null    uint8
 61  fuelsystem_mfi         205 non-null    uint8
 62  fuelsystem_mpfi        205 non-null    uint8
 63  fuelsystem_spdi        205 non-null    uint8
 64  fuelsystem_spfi        205 non-null    uint8
dtypes: float64(8), int64(7), uint8(50)
memory usage: 34.2 KB
```

Spliting the data into test and train

In [21]:
```
from sklearn.model_selection import train_test_split

df_train, df_test = train_test_split(auto1, train_size = 0.7, test_size = 0.3, ra
```

In [22]:
```
df_train.shape
```

Out[22]:
```
(143, 65)
```

In [23]:
```
df_test.shape
```

Out[23]:
```
(62, 65)
```

Rescaling the data:

In [24]:
```
auto_numeric.columns
```

Out[24]:
```
Index(['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight',
       'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price'],
      dtype='object')
```

In [25]:
```
col_list = ['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight',
       'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price']
```

In [26]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
```

In [27]:
```
df_train[col_list] = scaler.fit_transform(df_train[col_list])
```

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

Out[28]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight
count	1.430000e+02	1.430000e+02	1.430000e+02	1.430000e+02	1.430000e+02	1.430000e+02
mean	5.473477e-17	1.538785e-15	2.003060e-16	-4.093074e-15	5.450186e-16	-1.894367e-16
std	1.003515e+00	1.003515e+00	1.003515e+00	1.003515e+00	1.003515e+00	1.003515e+00
min	-2.347020e+00	-2.006930e+00	-2.574223e+00	-2.510760e+00	-2.371619e+00	-1.937401e+00
25%	-6.689008e-01	-6.771770e-01	-6.186702e-01	-8.565171e-01	-7.222984e-01	-7.711028e-01
50%	1.701590e-01	-3.405307e-01	-1.128552e-01	-1.993522e-01	6.112865e-02	-2.478347e-01
75%	1.701590e-01	4.505882e-01	7.076008e-01	4.804736e-01	7.414732e-01	7.203955e-01
max	1.848278e+00	2.874442e+00	2.324616e+00	2.927846e+00	2.287711e+00	2.812547e+00

8 rows × 65 columns

Model building:

In [29]:
```
y_train = df_train.pop('price')
X_train = df_train
```

Model building using RFE

In [30]:
```
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from sklearn.feature_selection import RFE
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

In [31]:
```
lr = LinearRegression()
lr.fit(X_train,y_train)

# Subsetting training data for 15 selected columns
rfe = RFE(lr,15)
rfe.fit(X_train, y_train)
```

Out[31]:
```
RFE(estimator=LinearRegression(), n_features_to_select=15)
```

In [32]:
```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[32]:
```
[('symboling', False, 48),
 ('wheelbase', False, 23),
 ('carlength', False, 22),
 ('carwidth', False, 8),
 ('carheight', False, 20),
 ('curbweight', False, 12),
 ('enginesize', True, 1),
 ('boreratio', False, 4),
 ('stroke', False, 11),
 ('compressionratio', False, 19),
 ('horsepower', False, 33),
 ('peakrpm', False, 31),
 ('citympg', False, 43),
 ('highwaympg', False, 39),
 ('fueltype_gas', False, 17),
 ('Company_audi', True, 1),
 ('Company_bmw', True, 1),
 ('Company_buick', True, 1),
 ('Company_chevrolet', False, 21),
 ('Company_dodge', False, 14),
 ('Company_honda', False, 15),
 ('Company_isuzu', False, 46),
 ('Company_jaguar', False, 30),
 ('Company_mazda', False, 35),
 ('Company_mercury', False, 47),
 ('Company_mitsubishi', False, 5),
 ('Company_nissan', False, 34),
 ('Company_peugeot', False, 7),
 ('Company_plymouth', False, 13),
 ('Company_porsche', True, 1),
 ('Company_renault', False, 42),
 ('Company_saab', True, 1),
 ('Company_subaru', False, 9),
 ('Company_toyota', False, 36),
 ('Company_volkswagen', False, 37),
 ('Company_volvo', True, 1),
 ('aspiration_turbo', False, 6),
 ('doornumber_two', False, 41),
 ('carbody_hardtop', False, 25),
 ('carbody_hatchback', False, 16),
 ('carbody_sedan', False, 26),
 ('carbody_wagon', False, 27),
 ('drivewheel_fwd', False, 44),
 ('drivewheel_rwd', False, 38),
 ('enginelocation_rear', True, 1),
 ('enginetype_dohcv', True, 1),
 ('enginetype_l', True, 1),
 ('enginetype_ohc', False, 45),
 ('enginetype_ohcf', False, 3),
 ('enginetype_ohcv', False, 32),
 ('enginetype_rotor', True, 1),
 ('cylindernumber_five', True, 1),
 ('cylindernumber_four', False, 2),
 ('cylindernumber_six', False, 10),
```

```
 ('cylindernumber_three', True, 1),
 ('cylindernumber_twelve', True, 1),
 ('cylindernumber_two', True, 1),
 ('fuelsystem_2bbl', False, 40),
 ('fuelsystem_4bbl', False, 24),
 ('fuelsystem_idi', False, 18),
 ('fuelsystem_mfi', False, 49),
 ('fuelsystem_mpfi', False, 29),
 ('fuelsystem_spdi', False, 28),
 ('fuelsystem_spfi', False, 50)]
```

In [33]:
```
cols = X_train.columns[rfe.support_]
cols
```

Out[33]:
```
Index(['enginesize', 'Company_audi', 'Company_bmw', 'Company_buick',
       'Company_porsche', 'Company_saab', 'Company_volvo',
       'enginelocation_rear', 'enginetype_dohcv', 'enginetype_l',
       'enginetype_rotor', 'cylindernumber_five', 'cylindernumber_three',
       'cylindernumber_twelve', 'cylindernumber_two'],
      dtype='object')
```

Model 1:

In [34]:
```
X1 = X_train[cols]
X1_sm = sm.add_constant(X1)

lr_1 = sm.OLS(y_train,X1_sm).fit()
```

In [35]:
```
print(lr_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.913
Model:                            OLS   Adj. R-squared:                  0.904
Method:                 Least Squares   F-statistic:                     96.25
Date:                Tue, 18 Jan 2022   Prob (F-statistic):           8.60e-61
Time:                        13:44:37   Log-Likelihood:                -28.111
No. Observations:                 143   AIC:                             86.22
Df Residuals:                     128   BIC:                             130.7
Df Model:                          14
Covariance Type:            nonrobust
================================================================================
==========
                            coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
const                    -0.2329      0.032     -7.195      0.000      -0.297
-0.169
enginesize                0.7369      0.038     19.522      0.000       0.662
0.812
Company_audi              0.6604      0.255      2.592      0.011       0.156
1.165
Company_bmw               1.2025      0.140      8.575      0.000       0.925
1.480
Company_buick             1.0544      0.210      5.016      0.000       0.639
1.470
Company_porsche           0.9148      0.315      2.908      0.004       0.292
1.537
Company_saab              0.5961      0.182      3.267      0.001       0.235
0.957
Company_volvo             0.6419      0.134      4.799      0.000       0.377
0.907
enginelocation_rear       0.7677      0.442      1.737      0.085      -0.107
1.642
enginetype_dohcv          0.2675      0.443      0.604      0.547      -0.609
1.144
enginetype_l              0.3211      0.116      2.762      0.007       0.091
0.551
enginetype_rotor          0.5875      0.082      7.185      0.000       0.426
0.749
cylindernumber_five       0.0999      0.211      0.473      0.637      -0.318
0.517
cylindernumber_three      0.0500      0.338      0.148      0.883      -0.618
0.718
cylindernumber_twelve    -0.4464      0.369     -1.211      0.228      -1.176
0.283
cylindernumber_two        0.5875      0.082      7.185      0.000       0.426
0.749
==============================================================================
Omnibus:                       16.668   Durbin-Watson:                   1.956
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               27.467
Skew:                           0.574   Prob(JB):                     1.09e-06
Kurtosis:                       4.814   Cond. No.                     7.04e+16
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The smallest eigenvalue is 2.96e-32. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

All the p- values are significant. Let us check VIF.

In [36]:

```
#VIF
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[36]:

	Features	VIF
10	enginetype_rotor	inf
14	cylindernumber_two	inf
11	cylindernumber_five	4.13
1	Company_audi	3.31
4	Company_porsche	3.02
3	Company_buick	2.16
7	enginelocation_rear	2.02
8	enginetype_dohcv	2.02
0	enginesize	1.93
13	cylindernumber_twelve	1.33
12	cylindernumber_three	1.17
9	enginetype_l	1.14
2	Company_bmw	1.10
6	Company_volvo	1.02
5	Company_saab	1.00

We see that there are a few variables which have an infinite/large VIF. These variables aren't of use. But manually elimination is time consuming and makes the code unnecessarily long. So let's try and build a model with 10 features this time using RFE.

Building the model with 10 variables:

In [37]:
```
lr2 = LinearRegression()

rfe2 = RFE(lr2,10)
rfe2.fit(X_train,y_train)
```

Out[37]:
```
RFE(estimator=LinearRegression(), n_features_to_select=10)
```

In [38]:
```
list(zip(X_train.columns,rfe2.support_,rfe2.ranking_))
```

Out[38]:
```
[('symboling', False, 53),
 ('wheelbase', False, 28),
 ('carlength', False, 27),
 ('carwidth', False, 13),
 ('carheight', False, 25),
 ('curbweight', False, 17),
 ('enginesize', True, 1),
 ('boreratio', False, 9),
 ('stroke', False, 16),
 ('compressionratio', False, 24),
 ('horsepower', False, 38),
 ('peakrpm', False, 36),
 ('citympg', False, 48),
 ('highwaympg', False, 44),
 ('fueltype_gas', False, 22),
 ('Company_audi', True, 1),
 ('Company_bmw', True, 1),
 ('Company_buick', True, 1),
 ('Company_chevrolet', False, 26),
 ('Company_dodge', False, 19),
 ('Company_honda', False, 20),
 ('Company_isuzu', False, 51),
 ('Company_jaguar', False, 35),
 ('Company_mazda', False, 40),
 ('Company_mercury', False, 52),
 ('Company_mitsubishi', False, 10),
 ('Company_nissan', False, 39),
 ('Company_peugeot', False, 12),
 ('Company_plymouth', False, 18),
 ('Company_porsche', True, 1),
 ('Company_renault', False, 47),
 ('Company_saab', True, 1),
 ('Company_subaru', False, 14),
 ('Company_toyota', False, 41),
 ('Company_volkswagen', False, 42),
 ('Company_volvo', True, 1),
 ('aspiration_turbo', False, 11),
 ('doornumber_two', False, 46),
 ('carbody_hardtop', False, 30),
 ('carbody_hatchback', False, 21),
 ('carbody_sedan', False, 31),
 ('carbody_wagon', False, 32),
 ('drivewheel_fwd', False, 49),
 ('drivewheel_rwd', False, 43),
 ('enginelocation_rear', True, 1),
 ('enginetype_dohcv', False, 4),
 ('enginetype_l', False, 3),
 ('enginetype_ohc', False, 50),
 ('enginetype_ohcf', False, 8),
 ('enginetype_ohcv', False, 37),
 ('enginetype_rotor', True, 1),
 ('cylindernumber_five', False, 5),
 ('cylindernumber_four', False, 7),
 ('cylindernumber_six', False, 15),
```

```
 ('cylindernumber_three', False, 6),
 ('cylindernumber_twelve', False, 2),
 ('cylindernumber_two', True, 1),
 ('fuelsystem_2bbl', False, 45),
 ('fuelsystem_4bbl', False, 29),
 ('fuelsystem_idi', False, 23),
 ('fuelsystem_mfi', False, 54),
 ('fuelsystem_mpfi', False, 34),
 ('fuelsystem_spdi', False, 33),
 ('fuelsystem_spfi', False, 55)]
```

In [39]:
```
supported_cols = X_train.columns[rfe2.support_]
supported_cols
```

Out[39]:
```
Index(['enginesize', 'Company_audi', 'Company_bmw', 'Company_buick',
       'Company_porsche', 'Company_saab', 'Company_volvo',
       'enginelocation_rear', 'enginetype_rotor', 'cylindernumber_two'],
      dtype='object')
```

Model 2:

In [40]:
```
X2 = X_train[supported_cols]
X2_sm = sm.add_constant(X2)

model_2 = sm.OLS(y_train,X2_sm).fit()
```

In [41]:
```
print(model_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.905
Model:                            OLS   Adj. R-squared:                  0.899
Method:                 Least Squares   F-statistic:                     141.3
Date:                Tue, 18 Jan 2022   Prob (F-statistic):           1.39e-63
Time:                        13:44:37   Log-Likelihood:                -34.380
No. Observations:                 143   AIC:                             88.76
Df Residuals:                     133   BIC:                             118.4
Df Model:                           9
Covariance Type:            nonrobust
===============================================================================
========
                          coef    std err          t      P>|t|      [0.025
0.975]
-------------------------------------------------------------------------------
--------
const                  -0.2121      0.031     -6.889      0.000      -0.273
-0.151
enginesize              0.7221      0.032     22.586      0.000       0.659
0.785
Company_audi            0.7427      0.146      5.073      0.000       0.453
1.032
Company_bmw             1.1980      0.140      8.573      0.000       0.922
1.474
Company_buick           1.1221      0.160      6.992      0.000       0.805
1.440
Company_porsche         1.0467      0.232      4.506      0.000       0.587
1.506
Company_saab            0.5739      0.187      3.074      0.003       0.205
0.943
Company_volvo           0.6284      0.135      4.640      0.000       0.361
0.896
enginelocation_rear     0.6401      0.391      1.637      0.104      -0.133
1.414
enginetype_rotor        0.5675      0.083      6.828      0.000       0.403
0.732
cylindernumber_two      0.5675      0.083      6.828      0.000       0.403
0.732
==============================================================================
Omnibus:                       16.950   Durbin-Watson:                   1.931
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               23.386
Skew:                           0.664   Prob(JB):                     8.35e-06
Kurtosis:                       4.471   Cond. No.                     3.34e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The smallest eigenvalue is 1.3e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [42]:
```
#VIF
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[42]:

	Features	VIF
8	enginetype_rotor	inf
9	cylindernumber_two	inf
4	Company_porsche	1.55
7	enginelocation_rear	1.50
0	enginesize	1.39
3	Company_buick	1.18
2	Company_bmw	1.07
6	Company_volvo	1.01
1	Company_audi	1.00
5	Company_saab	1.00

As we see, still there are columns with high VIF. Let us drop column -cylindernumber_two.

Model 3:

In [43]:
```
X3 = X2.drop(['cylindernumber_two'], axis =1)
X3_sm = sm.add_constant(X3)

Model_3 = sm.OLS(y_train,X3_sm).fit()
```

In [44]:
```
print(Model_3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.905
Model:                            OLS   Adj. R-squared:                  0.899
Method:                 Least Squares   F-statistic:                     141.3
Date:                Tue, 18 Jan 2022   Prob (F-statistic):           1.39e-63
Time:                        13:44:38   Log-Likelihood:                -34.380
No. Observations:                 143   AIC:                             88.76
Df Residuals:                     133   BIC:                             118.4
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
========
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const               -0.2121      0.031     -6.889      0.000      -0.273
-0.151
enginesize           0.7221      0.032     22.586      0.000       0.659
0.785
Company_audi         0.7427      0.146      5.073      0.000       0.453
1.032
Company_bmw          1.1980      0.140      8.573      0.000       0.922
1.474
Company_buick        1.1221      0.160      6.992      0.000       0.805
1.440
Company_porsche      1.0467      0.232      4.506      0.000       0.587
1.506
Company_saab         0.5739      0.187      3.074      0.003       0.205
0.943
Company_volvo        0.6284      0.135      4.640      0.000       0.361
0.896
enginelocation_rear  0.6401      0.391      1.637      0.104      -0.133
1.414
enginetype_rotor     1.1351      0.166      6.828      0.000       0.806
1.464
==============================================================================
Omnibus:                       16.950   Durbin-Watson:                   1.931
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               23.386
Skew:                           0.664   Prob(JB):                     8.35e-06
Kurtosis:                       4.471   Cond. No.                         15.7
==============================================================================

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

In [45]:

```
#VIF
vif = pd.DataFrame()
vif['Features'] = X3.columns
vif['VIF'] = [variance_inflation_factor(X3.values, i) for i in range(X3.shape[1])
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

Out[45]:

	Features	VIF
4	Company_porsche	1.55
7	enginelocation_rear	1.50
0	enginesize	1.39
3	Company_buick	1.18
2	Company_bmw	1.07
8	enginetype_rotor	1.06
6	Company_volvo	1.01
1	Company_audi	1.00
5	Company_saab	1.00

Model 4:

In [46]:

```
X4 = X3.drop(['enginelocation_rear'], axis =1)
X4_sm = sm.add_constant(X4)

Model_4 = sm.OLS(y_train,X4_sm).fit()
```

In [47]:
```
print(Model_4.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  price   R-squared:                       0.903
Model:                            OLS   Adj. R-squared:                  0.898
Method:                 Least Squares   F-statistic:                     156.6
Date:                Tue, 18 Jan 2022   Prob (F-statistic):           4.02e-64
Time:                        13:44:38   Log-Likelihood:                -35.806
No. Observations:                 143   AIC:                             89.61
Df Residuals:                     134   BIC:                             116.3
Df Model:                           8                                         
Covariance Type:            nonrobust                                         
================================================================================
=====
                       coef    std err          t      P>|t|      [0.025      
0.975]
--------------------------------------------------------------------------------
-----
const               -0.2118      0.031     -6.837      0.000      -0.273      -
0.150
enginesize           0.7238      0.032     22.514      0.000       0.660       
0.787
Company_audi         0.7420      0.147      5.036      0.000       0.451       
1.033
Company_bmw          1.1957      0.141      8.504      0.000       0.918       
1.474
Company_buick        1.1184      0.161      6.926      0.000       0.799       
1.438
Company_porsche      1.2572      0.195      6.459      0.000       0.872       
1.642
Company_saab         0.5737      0.188      3.054      0.003       0.202       
0.945
Company_volvo        0.6272      0.136      4.603      0.000       0.358       
0.897
enginetype_rotor     1.1370      0.167      6.797      0.000       0.806       
1.468
==============================================================================
Omnibus:                       15.610   Durbin-Watson:                   1.961
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               20.300
Skew:                           0.644   Prob(JB):                     3.91e-05
Kurtosis:                       4.322   Cond. No.                         7.67
==============================================================================

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

In [48]:
```
#VIF
vif = pd.DataFrame()
vif['Features'] = X4.columns
vif['VIF'] = [variance_inflation_factor(X4.values, i) for i in range(X4.shape[1])
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

Out[48]:

	Features	VIF
0	enginesize	1.39
3	Company_buick	1.18
2	Company_bmw	1.07
4	Company_porsche	1.06
7	enginetype_rotor	1.06
6	Company_volvo	1.01
1	Company_audi	1.00
5	Company_saab	1.00

All the VIF values and p-values seem to be in a good range. Also the Adjusted R-squared is 89%. This model is explaining most of the variance without being too complex.

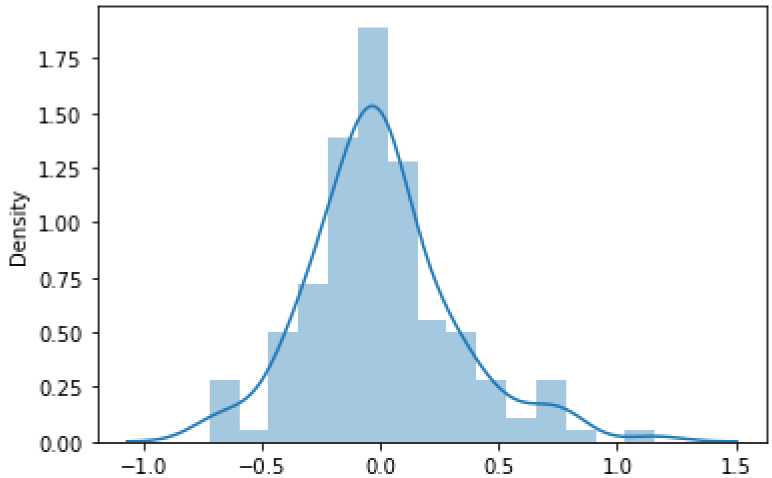
Residual analysis:

In [49]:
```
y_train_pred = Model_4.predict(X4_sm)
y_train_pred.head()
```

Out[49]:
```
122   -0.689662
125    1.507146
166   -0.689662
1     -0.122374
199    0.504873
dtype: float64
```

In [50]:
```
Residual = y_train- y_train_pred
sns.distplot(Residual, bins =15)
```

Out[50]: <AxesSubplot:ylabel='Density'>



Error term is normally distributed.

Making Predictions:

In [51]:
```
df_test[col_list] = scaler.transform(df_test[col_list])
y_test = df_test.pop('price')
X_test = df_test
final_cols = X4.columns
X_test_model4= X_test[final_cols]
X_test_model4.head()
```

Out[51]:

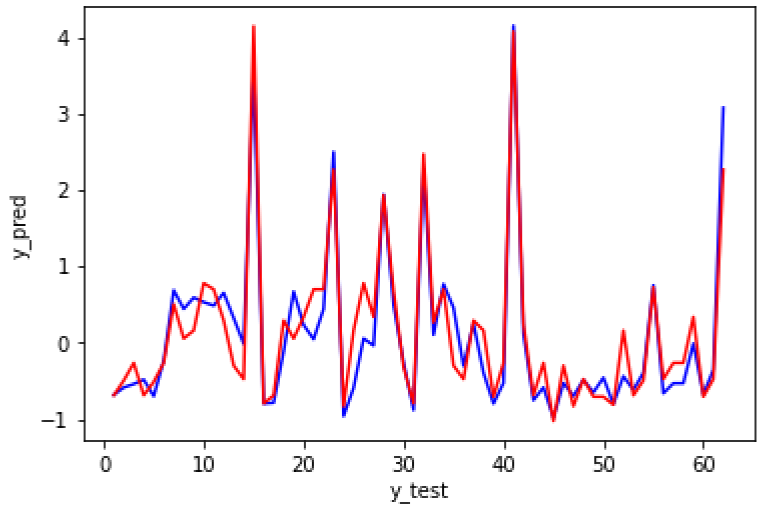
	enginesize	Company_audi	Company_bmw	Company_buick	Company_porsche	Company_saa
160	-0.660242	0	0	0	0	
186	-0.390836	0	0	0	0	
59	-0.072447	0	0	0	0	
165	-0.660242	0	0	0	0	
140	-0.415328	0	0	0	0	

In [52]:
```
X_test_sm = sm.add_constant(X_test_model4)
y_pred = Model_4.predict(X_test_sm)
y_pred.head()
```

Out[52]:
```
160   -0.689662
186   -0.494657
59    -0.264196
165   -0.689662
140   -0.512384
dtype: float64
```

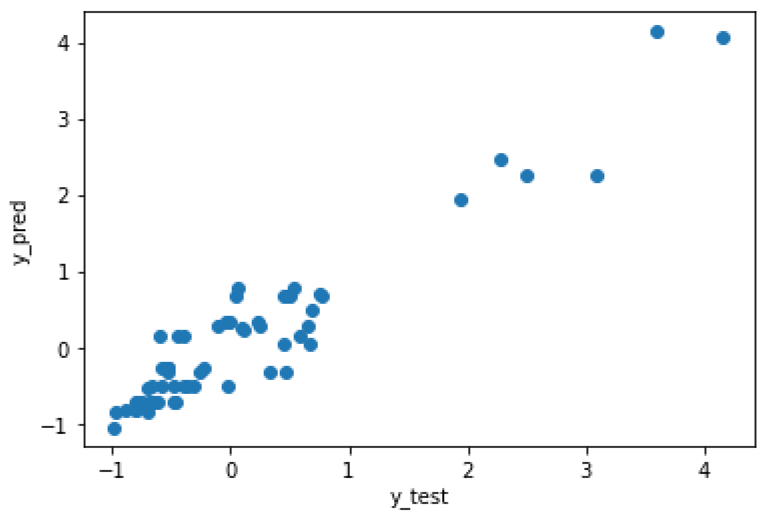
In [53]:
```
c = [i for i in range(1,63,1)]
plt.plot(c, y_test,color = 'Blue')
plt.plot(c, y_pred,color = 'red')
plt.xlabel('y_test')
plt.ylabel('y_pred')
```

Out[53]: Text(0, 0.5, 'y_pred')



In [54]:
```
plt.scatter(y_test, y_pred)
plt.xlabel('y_test')
plt.ylabel('y_pred')
```

Out[54]: Text(0, 0.5, 'y_pred')



Though the model is doing good at the beginning, still there are few high values which model is not able to explain.

Evaluation:

In [55]:
```
r_squ = r2_score(y_test,y_pred)
r_squ
```

Out[55]: 0.9053256288000193

So linear equation for price can be given as:
price = -0.2118 + enginesize*0.7238* + *Company_audi*0.7420 +Company_bmw*1.1957*
+ *Company_buick*1.1184 + Company_porsche*1.2572* + *Company_saab**0.5737 +
Company_volvo*0.6272* + *enginetype_rotor**1.1370

These are the variables that are significant in predicting the price of a car.

- enginesize
- Company_audi
- Company_bmw
- Company_buick
- Company_porsche
- Company_saab
- Company_volvo
- enginetype_rotor

In [ ]: