ML-1 Graded Assessment

DATA_SET:

A Chinese automobile company Geely Auto 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

Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the Americal market.

- Car_ID== Unique id of each observation (Interger)
- Symboling== Its assigned insurance risk rating, A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.(Categorical)
- carCompany== Name of car company (Categorical)
- fueltype== Car fuel type i.e gas or diesel (Categorical)
- aspiration== Aspiration used in a car (Categorical)
- doornumber== Number of doors in a car (Categorical)
- carbody== body of car (Categorical)
- drivewheel== type of drive wheel (Categorical)
- enginelocation== Location of car engine (Categorical)
- wheelbase== Weelbase of car (Numeric)
- carlength== Length of car (Numeric)
- carwidth== Width of car (Numeric)
- carheight== height of car (Numeric)
- curbweight== The weight of a car without occupants or baggage. (Numeric)
- enginetype== Type of engine. (Categorical)
- cylindernumber== cylinder placed in the car (Categorical)
- enginesize== Size of car (Numeric)
- fuelsystem== Fuel system of car (Categorical)
- boreratio== Boreratio of car (Numeric)
- stroke== Stroke or volume inside the engine (Numeric)
- compressionratio== compression ratio of car (Numeric)

- horsepower== Horsepower (Numeric)
- peakrpm== car peak rpm (Numeric)
- citympg== Mileage in city (Numeric)
- highwaympg== Mileage on highway (Numeric)
- price(Dependent variable)== Price of car (Numeric)

In [27]:
```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1. Data Understanding (5 marks)

a. Read the dataset (tab, csv, xls, txt, inbuilt dataset). What are the number of rows and no. of cols & types of variables (continuous, categorical etc.)? (1 MARK)

b. Calculate five-point summary for numerical variables (1 MARK)

c. Summarize observations for categorical variables – no. of categories, % observations in each category. (1 mark)

d. Check for defects in the data such as missing values, null, outliers, etc. (2 marks)

a. Read the dataset (tab, csv, xls, txt, inbuilt dataset). What are the number of rows and no. of cols & types of variables (continuous, categorical etc.)? (1 MARK)

In [29]:
```
cars = pd.read_csv('Car_Data.csv')
cars.head()
```

Out[29]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	bore
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	bore
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	

5 rows × 26 columns

In [30]:
```
cars.shape
```

Out[30]: (205, 26)

In [31]:
```
cat_cols = cars.select_dtypes(include = 'object')
num_cols = cars.select_dtypes(include = np.number)
print('Continuous variables are : ',num_cols.columns)
print('Categorical Columns are : ',cat_cols.columns)
```

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

b. Calculate five-point summary for numerical variables (1 MARK)

In [32]:
```
cars.describe()
```

Out[32]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsep
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.00
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	10.142537	104.11
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	3.972040	39.54
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.00
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	8.600000	70.00

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsep
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.00
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	9.400000	116.00
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	288.00

c. Summarize observations for categorical variables – no. of categories, % observations in each category. (1 mark)

In [33]:
```
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 [34]:
```
print('% Values in each categorical columns')
for i in cat_cols.columns:
    print('\n% Values in column ',i)
    print((cars[i].value_counts()/len(cars[i])*100))
```

```
% Values in each categorical columns

% Values in column  CarName
peugeot 504                2.926829
toyota corolla             2.926829
toyota corona              2.926829
subaru dl                  1.951220
mitsubishi outlander       1.463415
                             ...
buick skyhawk              0.487805
subaru tribeca             0.487805
buick electra 225 custom   0.487805
alfa-romero stelvio        0.487805
volkswagen rabbit          0.487805
Name: CarName, Length: 147, dtype: float64

% Values in column  fueltype
gas       90.243902
diesel     9.756098
Name: fueltype, dtype: float64

% Values in column  aspiration
std      81.95122
turbo    18.04878
Name: aspiration, dtype: float64

% Values in column  doornumber
four    56.097561
two     43.902439
Name: doornumber, dtype: float64

% Values in column  carbody
sedan          46.829268
hatchback      34.146341
wagon          12.195122
hardtop         3.902439
convertible     2.926829
Name: carbody, dtype: float64
```

```
% Values in column  drivewheel
fwd    58.536585
rwd    37.073171
4wd     4.390244
Name: drivewheel, dtype: float64

% Values in column  enginelocation
front    98.536585
rear      1.463415
Name: enginelocation, dtype: float64

% Values in column  enginetype
ohc      72.195122
ohcf      7.317073
ohcv      6.341463
dohc      5.853659
l         5.853659
rotor     1.951220
dohcv     0.487805
Name: enginetype, dtype: float64

% Values in column  cylindernumber
four      77.560976
six       11.707317
five       5.365854
eight      2.439024
two        1.951220
twelve     0.487805
three      0.487805
Name: cylindernumber, dtype: float64

% Values in column  fuelsystem
mpfi    45.853659
2bbl    32.195122
idi      9.756098
1bbl     5.365854
spdi     4.390244
4bbl     1.463415
mfi      0.487805
spfi     0.487805
Name: fuelsystem, dtype: float64
```

In [35]:

```
print('% Values in each categorical columns')
for i in cat_cols.columns:
    print('\n% Values in column ',i)
    print((cars[i].value_counts(1)))
```

```
% Values in each categorical columns

% Values in column  CarName
peugeot 504                 0.029268
toyota corolla              0.029268
toyota corona               0.029268
subaru dl                   0.019512
mitsubishi outlander        0.014634
                              ...   
buick skyhawk               0.004878
subaru tribeca              0.004878
buick electra 225 custom    0.004878
alfa-romero stelvio         0.004878
volkswagen rabbit           0.004878
Name: CarName, Length: 147, dtype: float64

% Values in column  fueltype
gas       0.902439
diesel    0.097561
Name: fueltype, dtype: float64

% Values in column  aspiration
std      0.819512
turbo    0.180488
Name: aspiration, dtype: float64

% Values in column  doornumber
four    0.560976
two     0.439024
Name: doornumber, dtype: float64

% Values in column  carbody
sedan          0.468293
hatchback      0.341463
wagon          0.121951
hardtop        0.039024
convertible    0.029268
Name: carbody, dtype: float64

% Values in column  drivewheel
fwd    0.585366
rwd    0.370732
4wd    0.043902
Name: drivewheel, dtype: float64

% Values in column  enginelocation
front    0.985366
```

```
rear     0.014634
Name: enginelocation, dtype: float64

% Values in column  enginetype
ohc      0.721951
ohcf     0.073171
ohcv     0.063415
dohc     0.058537
l        0.058537
rotor    0.019512
dohcv    0.004878
Name: enginetype, dtype: float64

% Values in column  cylindernumber
four      0.775610
six       0.117073
five      0.053659
eight     0.024390
two       0.019512
twelve    0.004878
three     0.004878
Name: cylindernumber, dtype: float64

% Values in column  fuelsystem
mpfi    0.458537
2bbl    0.321951
idi     0.097561
1bbl    0.053659
spdi    0.043902
4bbl    0.014634
mfi     0.004878
spfi    0.004878
Name: fuelsystem, dtype: float64
```

d. Check for defects in the data such as missing values, null, outliers, etc. (2 marks)

In [36]:
```
cars.isnull().sum()
```

Out[36]:
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
```

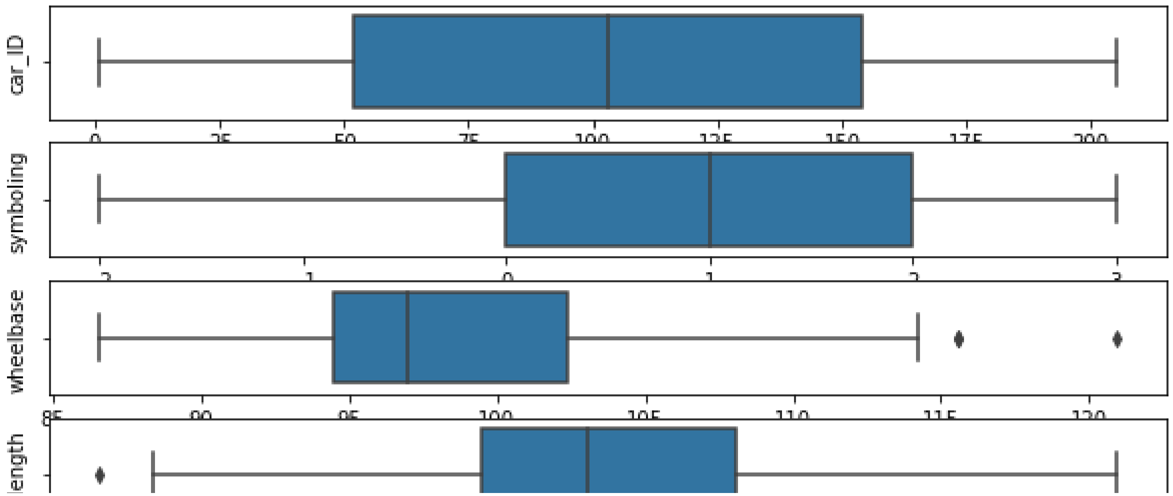
```
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

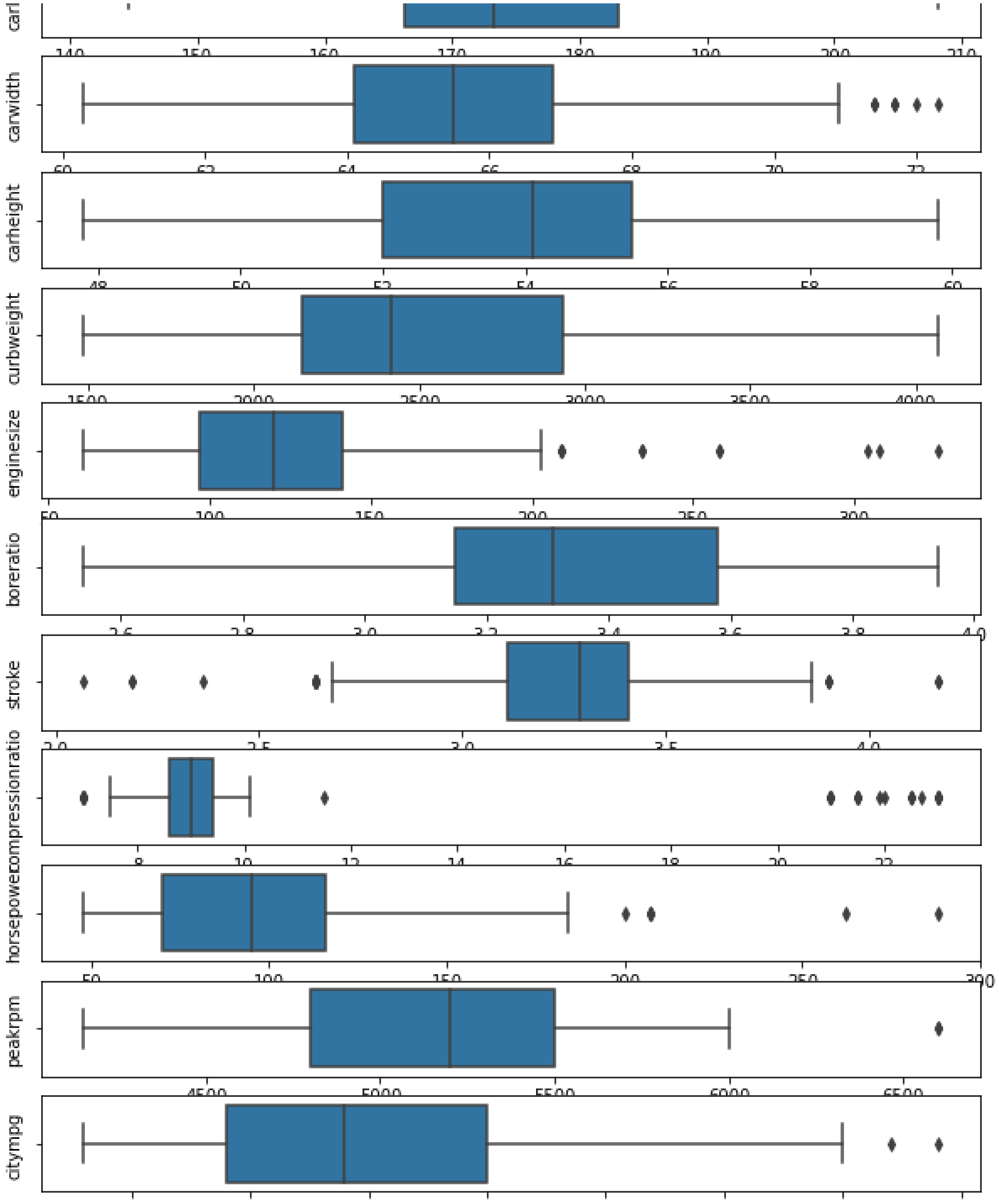
- As we can see, this dataset does not contain any missing value

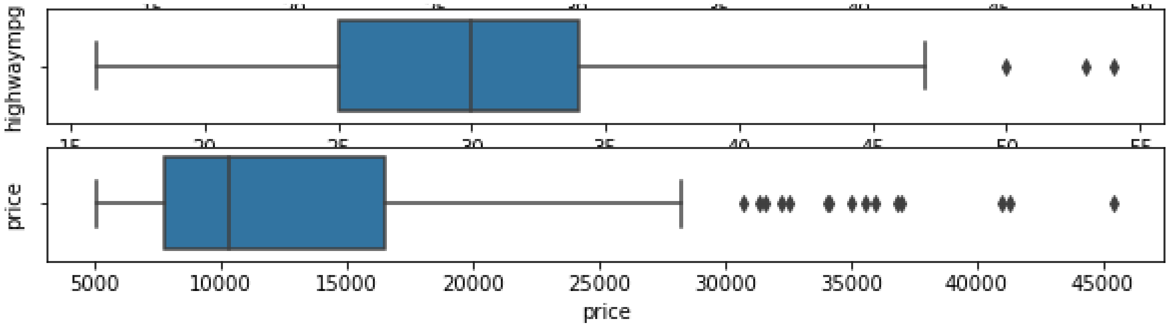
*** Outliers:**

In [37]:

```
plt.figure(figsize=(10,20))
for i,col in enumerate(num_cols,1):
    plt.subplot(16,1,i)
    sns.boxplot(cars[col])
    plt.ylabel(col)
plt.show()
```




carl
carwidth
carheight
curbweight
enginesize
boreratio
stroke
compressionratio
horsepower
peakrpm
citympg
4.0
300



Data Cleaning

In [38]:
```
cars['CarName'].unique()
```

Out[38]:
```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
       'dodge rampage', 'dodge challenger se', 'dodge d200',
       'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
       'dodge coronet custom', 'dodge dart custom',
       'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
       'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
       'honda accord', 'honda civic 1300', 'honda prelude',
       'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max',
       'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
       'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
       'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
       'mazda glc 4', 'mazda glc custom l', 'mazda glc custom',
       'buick electra 225 custom', 'buick century luxus (sw)',
       'buick century', 'buick skyhawk', 'buick opel isuzu deluxe',
       'buick skylark', 'buick century special',
       'buick regal sport coupe (turbo)', 'mercury cougar',
       'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander',
       'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero',
       'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue',
       'nissan latio', 'nissan titan', 'nissan leaf', 'nissan juke',
       'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz',
       'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks',
       'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl',
       'peugeot 505s turbo diesel', 'plymouth fury iii',
       'plymouth cricket', 'plymouth satellite custom (sw)',
       'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster',
```

```
       'porsche macan', 'porcshce panamera', 'porsche cayenne',
       'porsche boxter', 'renault 12tl', 'renault 5 gtl', 'saab 99e',
       'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz',
       'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',
       'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
       'toyota corolla 1200', 'toyota corona hardtop',
       'toyota corolla 1600 (sw)', 'toyota carina', 'toyota mark ii',
       'toyota corolla', 'toyota corolla liftback',
       'toyota celica gt liftback', 'toyota corolla tercel',
       'toyota corona liftback', 'toyota starlet', 'toyota tercel',
       'toyota cressida', 'toyota celica gt', 'toyouta tercel',
       'vokswagen rabbit', 'volkswagen 1131 deluxe sedan',
       'volkswagen model 111', 'volkswagen type 3', 'volkswagen 411 (sw)',
       'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
       'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
       'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
       'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
```

In [39]:

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

Out[39]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	b
0	1	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	
3	4	2	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	
4	5	2	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	

5 rows × 26 columns

In [40]:

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

Out[40]:

```
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)
```

Fixing invalid values

- There seems to be some spelling error in the CompanyName column.
    - `maxda = mazda`
    - `Nissan = nissan`
    - `porsche = porcshce`
    - `toyota = toyouta`
    - `vokswagen = volkswagen = vw`

In [41]:

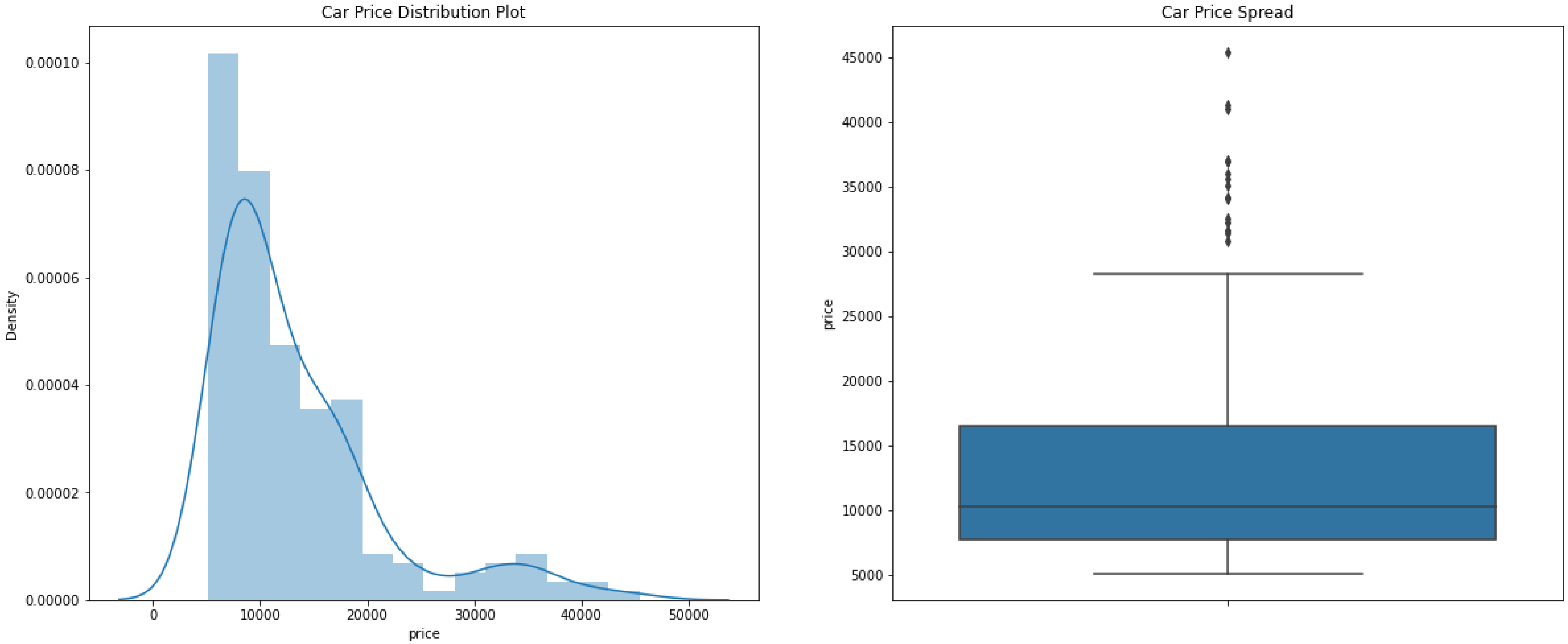
```
cars.CompanyName = cars.CompanyName.str.lower()

def replace_name(a,b):
    cars.CompanyName.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')

cars.CompanyName.unique()
```

Out[41]:

```
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 [42]:

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

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

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

plt.show()
```



In [43]:
```
print(cars.price.describe(percentiles = [0.25,0.50,0.75,0.85,0.90,1]))
```

```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
25%       7788.000000
50%      10295.000000
75%      16503.000000
85%      18500.000000
90%      22563.000000
100%     45400.000000
max      45400.000000
Name: price, dtype: float64
```

Inference :

1. The plot seemed to be right-skewed, meaning that the most prices in the dataset are low(Below 15,000).
2. There is a significant difference between the mean and the median of the price distribution.

3. 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.)

2. Data Preparation (15 marks)

a. Fix the defects found above and do appropriate treatment if any. (5 marks)

b. Visualize the data using relevant plots. Find out the variables which are highly correlated with target variable? (5 marks)

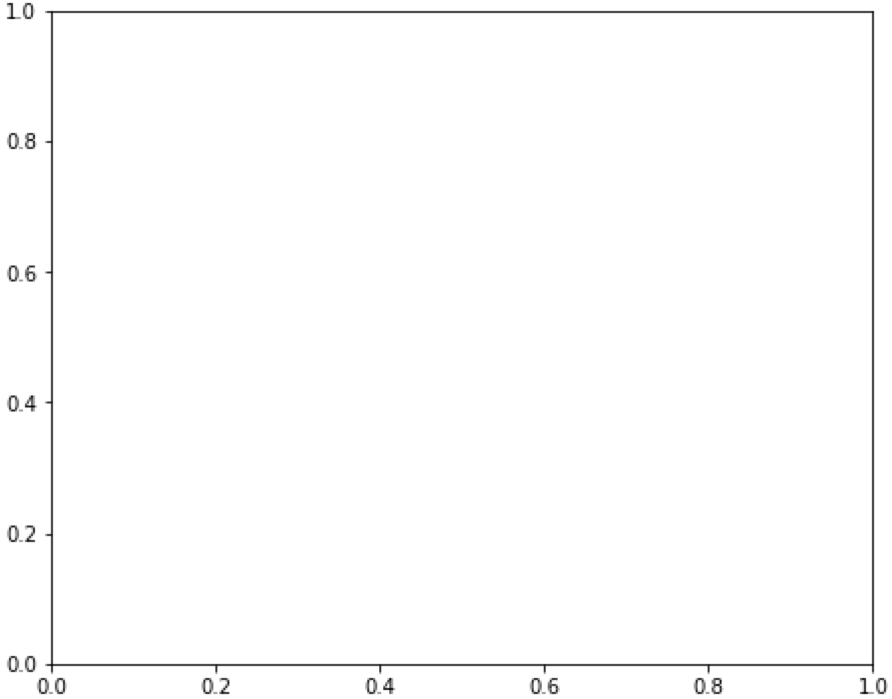
c. Do you want to exclude some variables from the model based on this analysis? What other actions will you take? (2 marks)

d. Split dataset into train and test (70:30). Are both train and test representative of the overall data? How would you ascertain this statistically? (3 marks)

In [44]:
```
plt.figure(figsize=(25, 6))

plt.subplot(1,3,1)
plt1 = cars.CompanyName.value_counts().plot('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('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('bar')
plt.title('Car Type Histogram')
plt1.set(xlabel = 'Car Type', ylabel='Frequency of Car type')

plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-44-63515b9191a3> in <module>
      2 
      3 plt.subplot(1,3,1)
----> 4 plt1 = cars.CompanyName.value_counts().plot('bar')
      5 plt.title('Companies Histogram')
      6 plt1.set(xlabel = 'Car company', ylabel='Frequency of company')
```

```
~\anaconda3\lib\site-packages\pandas\plotting\_core.py in __call__(self, *args, **kwargs)
    875         plot_backend = _get_plot_backend(kwargs.pop("backend", None))
    876
--> 877         x, y, kind, kwargs = self._get_call_args(
    878             plot_backend.__name__, self._parent, args, kwargs
    879         )

~\anaconda3\lib\site-packages\pandas\plotting\_core.py in _get_call_args(backend_name, data, args, kwargs)
    859                 f"`Series.plot({positional_args})`."
    860             )
--> 861             raise TypeError(msg)
    862
    863         pos_args = {name: value for value, (name, _) in zip(args, arg_def)}

TypeError: `Series.plot()` should not be called with positional arguments, only keyword arguments. The order of positional argumen
ts will change in the future. Use `Series.plot(kind='bar')` instead of `Series.plot('bar',)`.
```

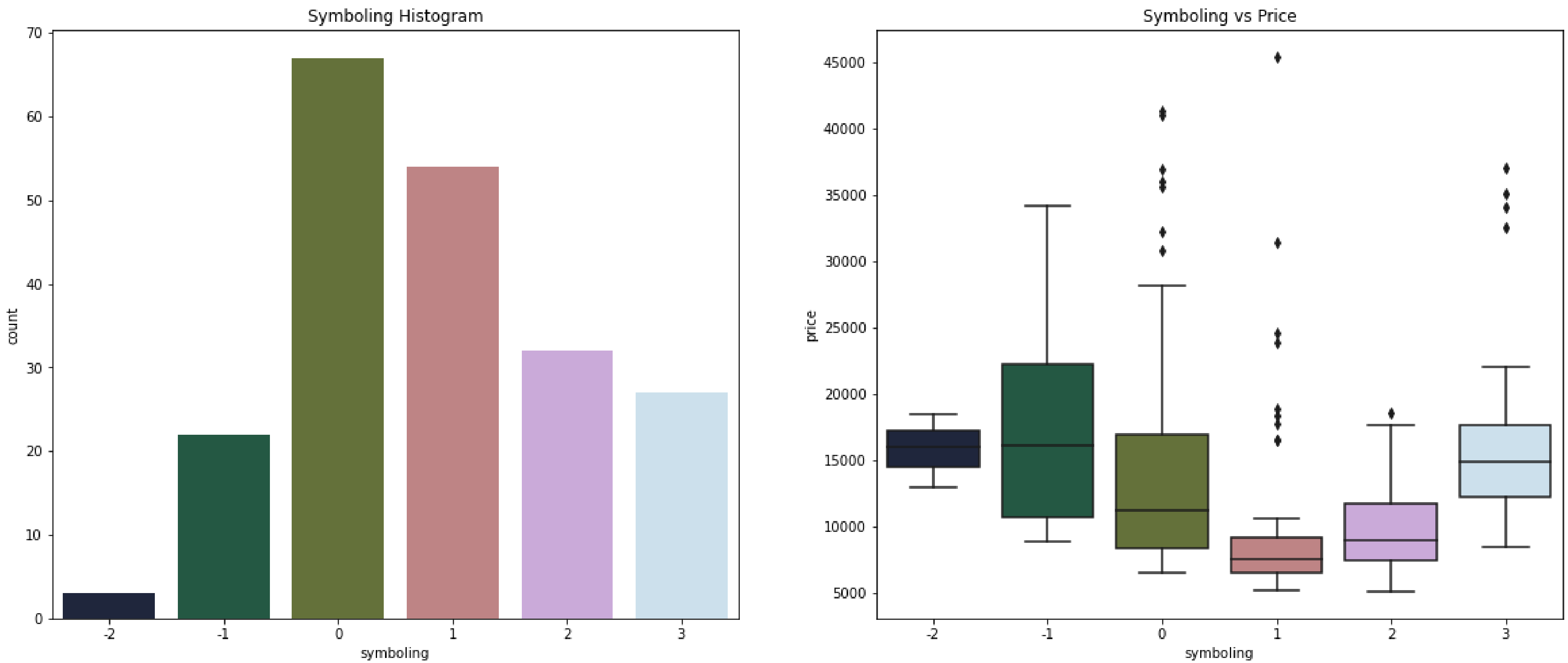


Inference :

1. `Toyota` seemed to be favored car company.
2. Number of `gas` fueled cars are more than `diesel` .

3. `sedan` is the top car type prefered.

In [45]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Symboling Histogram')
sns.countplot(cars.symboling, palette=("cubehelix"))

plt.subplot(1,2,2)
plt.title('Symboling vs Price')
sns.boxplot(x=cars.symboling, y=cars.price, palette=("cubehelix"))

plt.show()
```

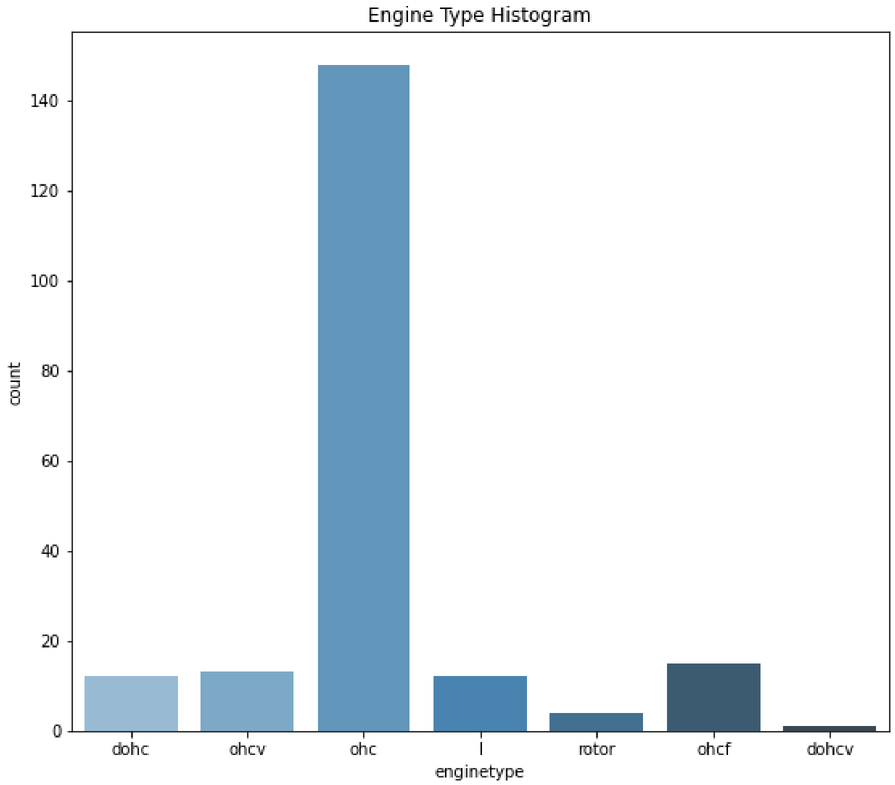


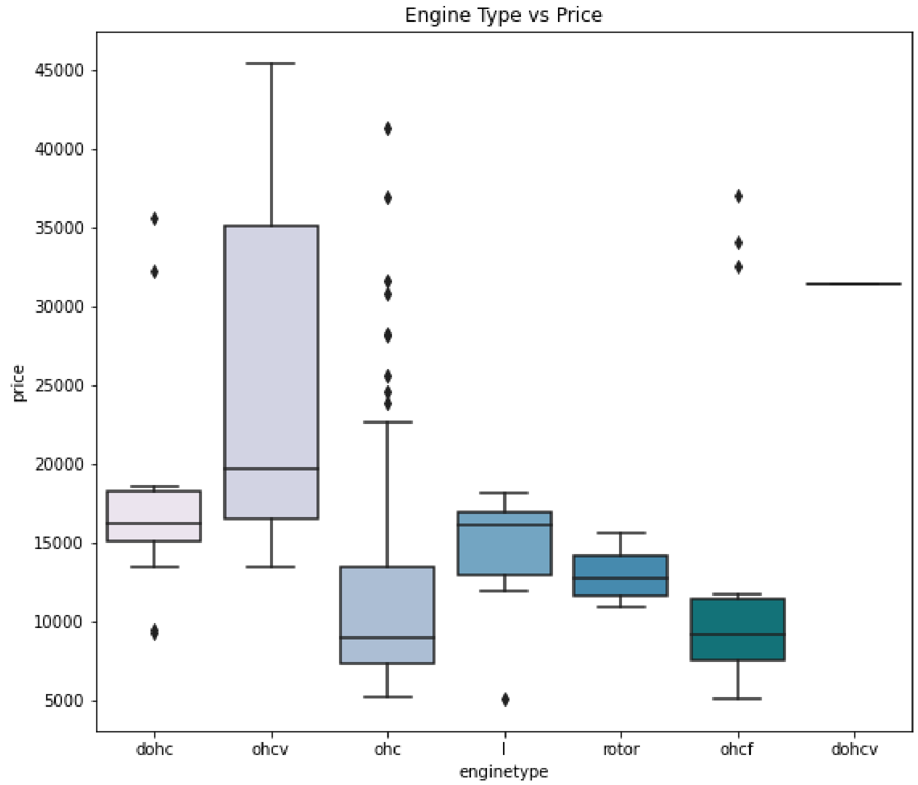
Inference :

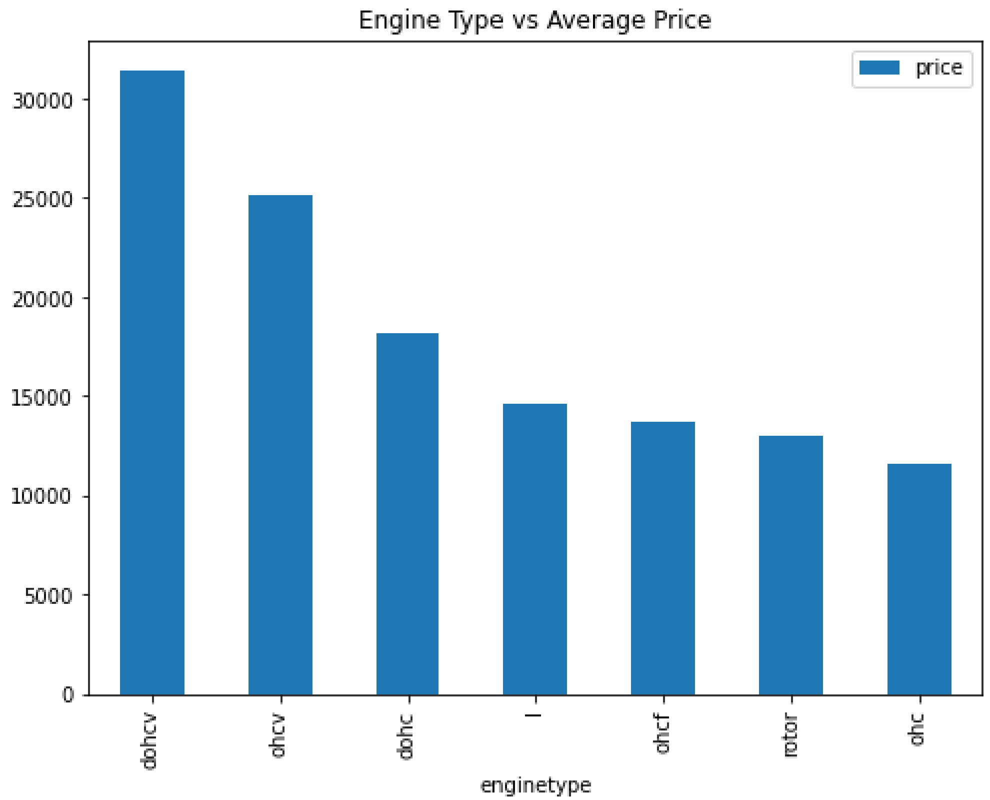
1. It seems that the symboling with `0` and `1` values have high number of rows (i.e. They are most sold.)

2. The cars with `-1` symboling seems to be high priced (as it makes sense too, insurance risk rating -1 is quite good). But it seems that symboling with `3` value has the price range similar to `-2` value. There is a dip in price at symboling `1`.

In [46]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Engine Type Histogram')
sns.countplot(cars.enginetype, palette=("Blues_d"))

plt.subplot(1,2,2)
plt.title('Engine Type vs Price')
sns.boxplot(x=cars.enginetype, y=cars.price, palette=("PuBuGn"))

plt.show()

df = pd.DataFrame(cars.groupby(['enginetype'])['price'].mean().sort_values(ascending = False))
df.plot.bar(figsize=(8,6))
plt.title('Engine Type vs Average Price')
plt.show()
```


Engine Type Histogram
count
0
20
40
60
80
100
120
140
dohc
ohcv
ohc
l
rotor
ohcf
dohcv
enginetype


Engine Type vs Price
price
5000
10000
15000
20000
25000
30000
35000
40000
45000
dohc
ohcv
ohc
l
rotor
ohcf
dohcv
enginetype



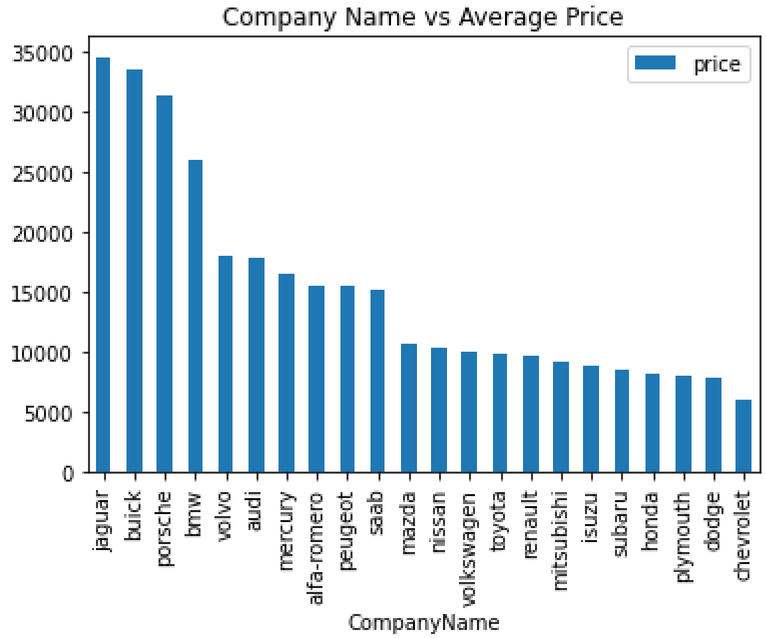
Inference :

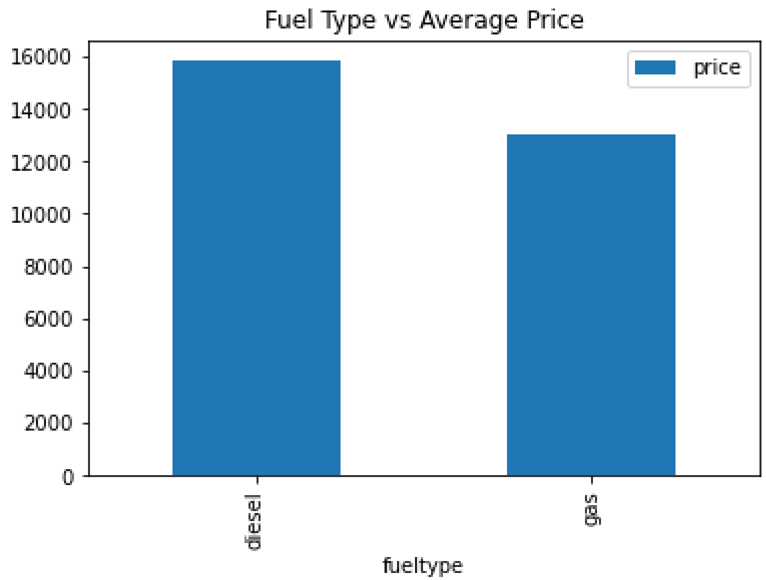
1. `ohc` Engine type seems to be most favored type.
2. `ohcv` has the highest price range (While `dohcv` has only one row), `ohc` and `ohcf` have the low price range.

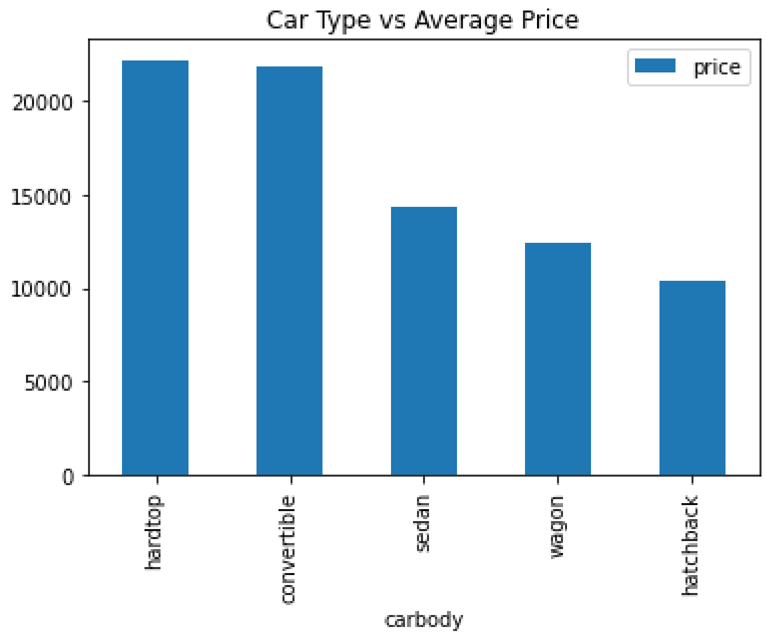
In [47]:
```
plt.figure(figsize=(25, 6))

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

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

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

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





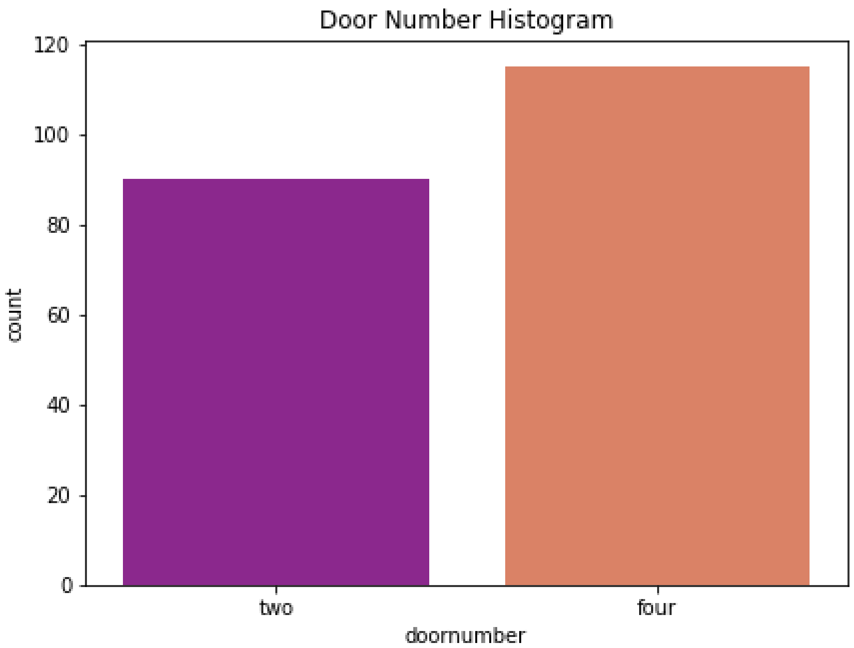


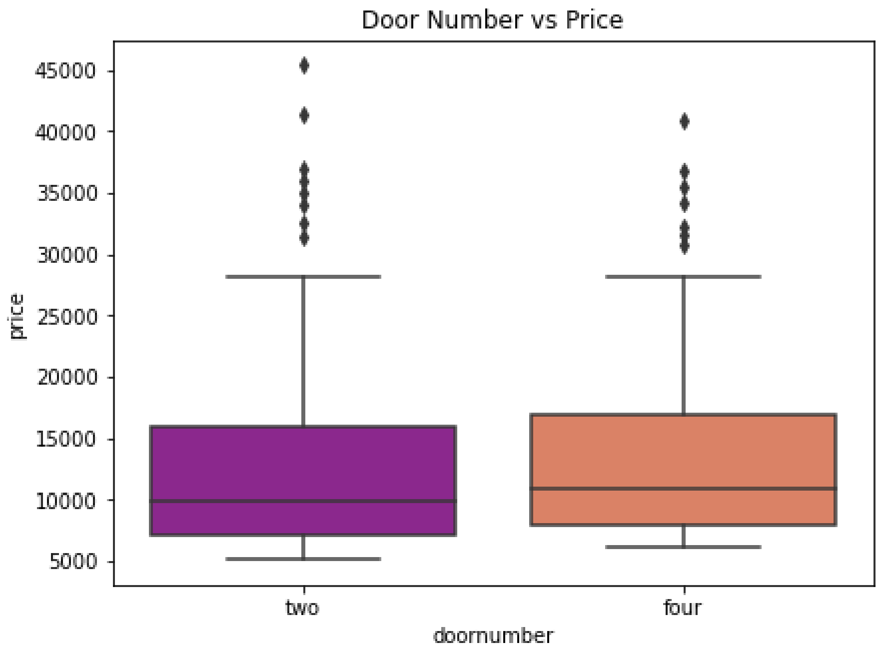
Inference :

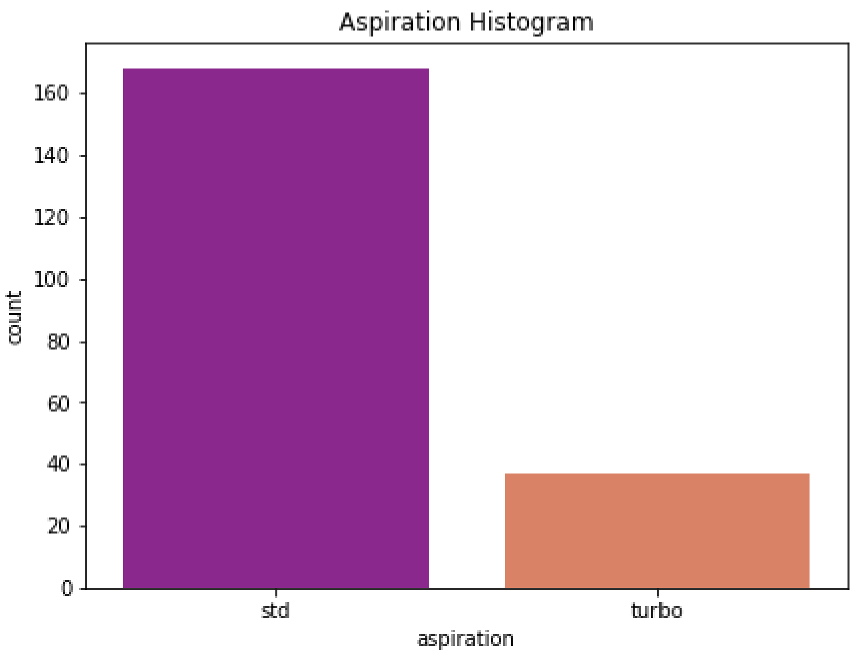
1. `Jaguar` and `Buick` seem to have highest average price.

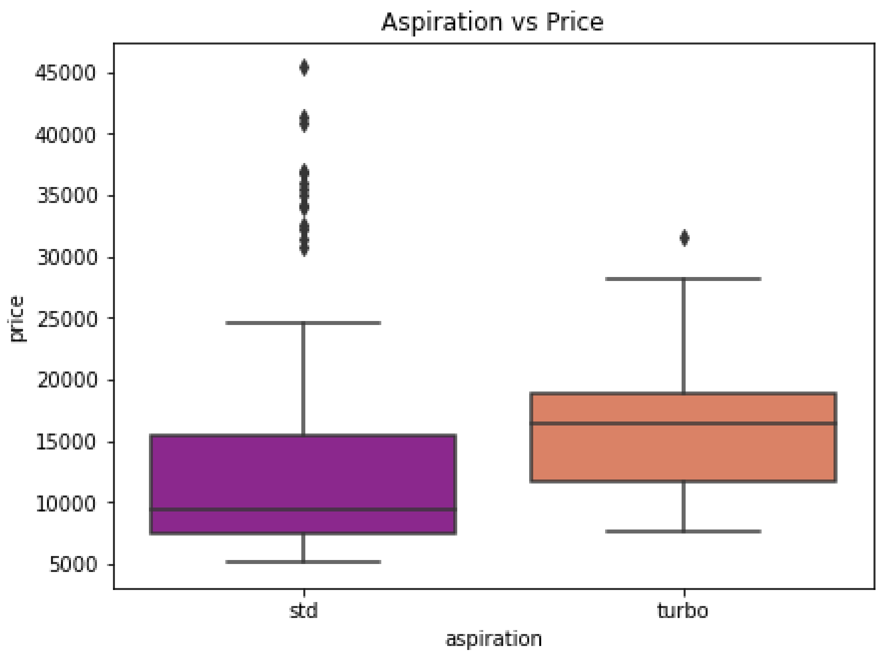
2. `diesel` has higher average price than gas.
3. `hardtop` and `convertible` have higher average price.

In [48]:
```
plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
plt.title('Door Number Histogram')
sns.countplot(cars.doornumber, palette=("plasma"))

plt.subplot(1,2,2)
plt.title('Door Number vs Price')
sns.boxplot(x=cars.doornumber, y=cars.price, palette=("plasma"))

plt.show()

plt.figure(figsize=(15,5))

plt.subplot(1,2,1)
plt.title('Aspiration Histogram')
sns.countplot(cars.aspiration, palette=("plasma"))

plt.subplot(1,2,2)
plt.title('Aspiration vs Price')
sns.boxplot(x=cars.aspiration, y=cars.price, palette=("plasma"))

plt.show()
```


Door Number Histogram
count
120
100
80
60
40
20
0
two
four
doornumber


Door Number vs Price
price
45000
40000
35000
30000
25000
20000
15000
10000
5000
two
four
doornumber

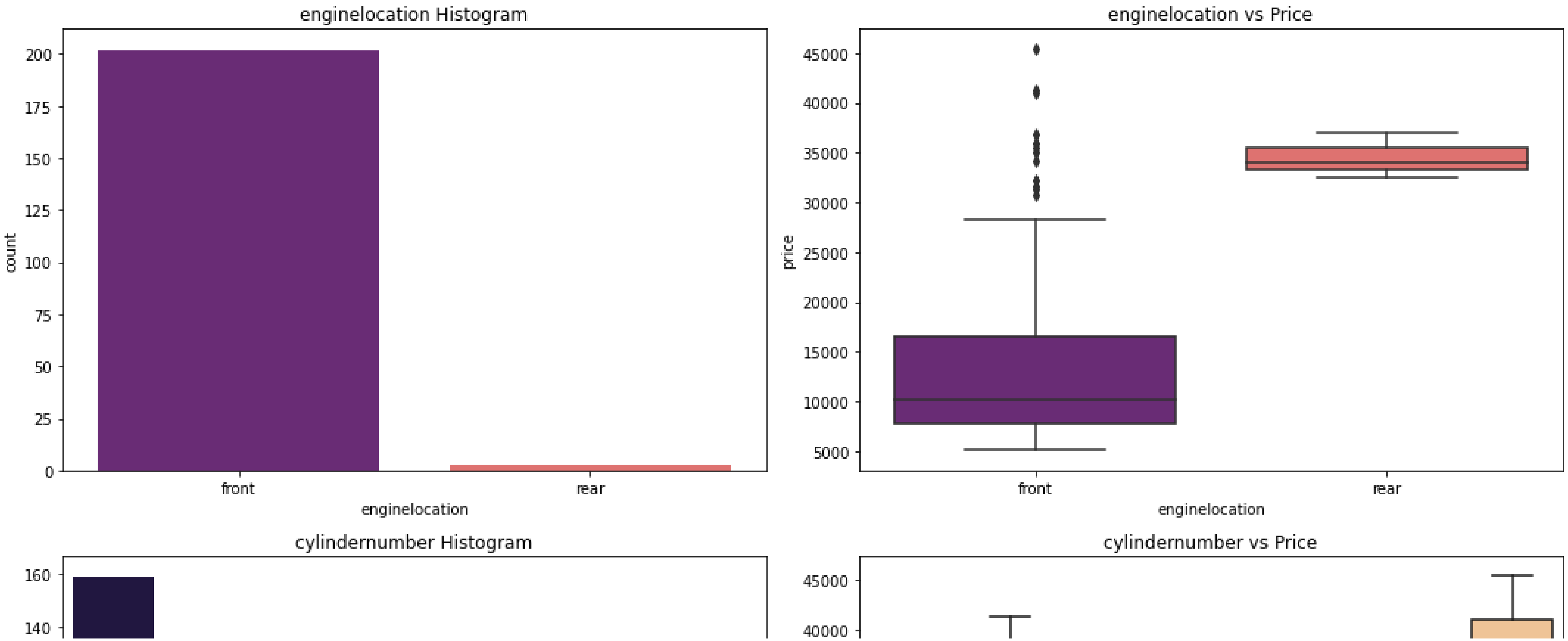

Aspiration Histogram
count
160
140
120
100
80
60
40
20
0
std
turbo
aspiration

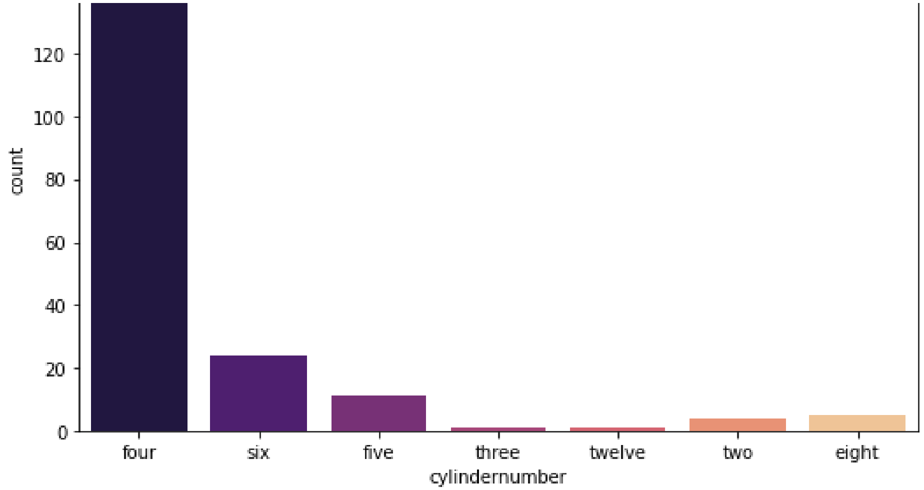

Aspiration vs Price
price
45000
40000
35000
30000
25000
20000
15000
10000
5000
std
turbo
aspiration

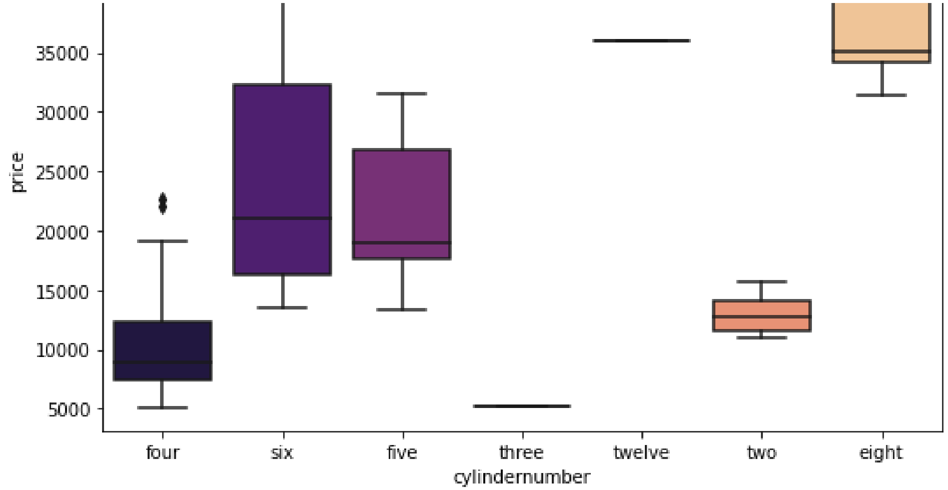
Inference :

1. `doornumber` variable is not affacting the price much. There is no significant difference between the categories in it.
2. It seems aspiration with `turbo` have higher price range than the `std` (though it has some high values outside the whiskers.)

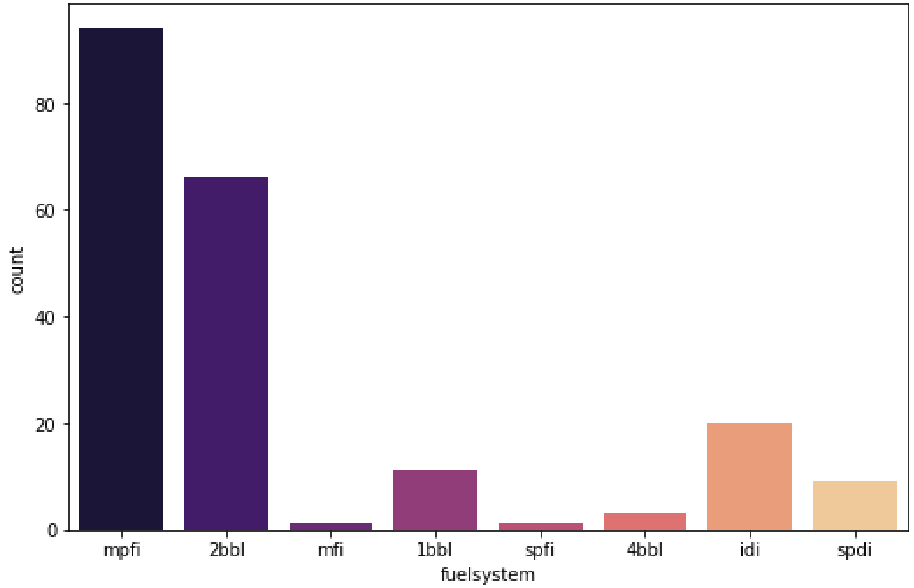
In [49]:
```
def plot_count(x,fig):
    plt.subplot(4,2,fig)
    plt.title(x+' Histogram')
    sns.countplot(cars[x],palette=("magma"))
    plt.subplot(4,2,(fig+1))
    plt.title(x+' vs Price')
    sns.boxplot(x=cars[x], y=cars.price, palette=("magma"))

plt.figure(figsize=(15,20))

plot_count('enginelocation', 1)
plot_count('cylindernumber', 3)
plot_count('fuelsystem', 5)
plot_count('drivewheel', 7)

plt.tight_layout()
```



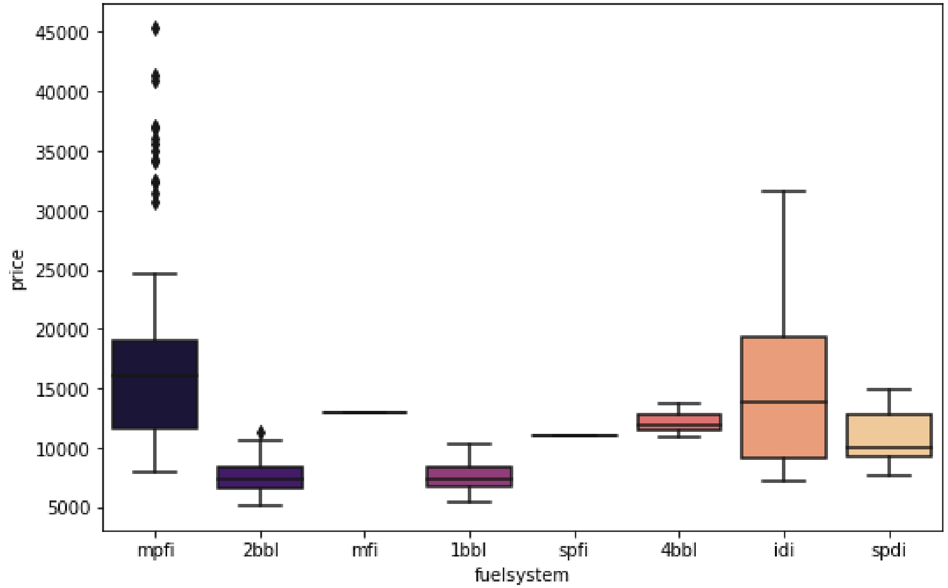




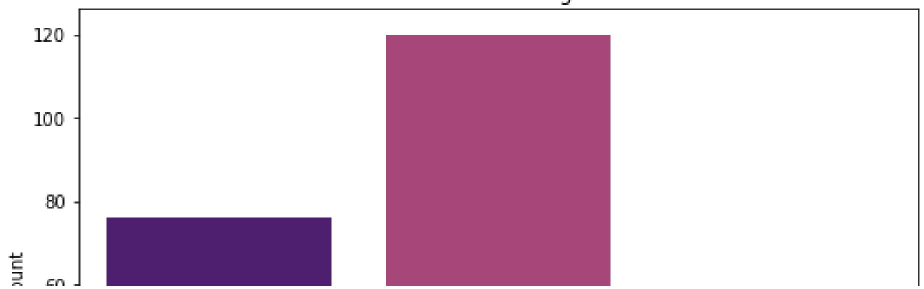
fuelsystem Histogram



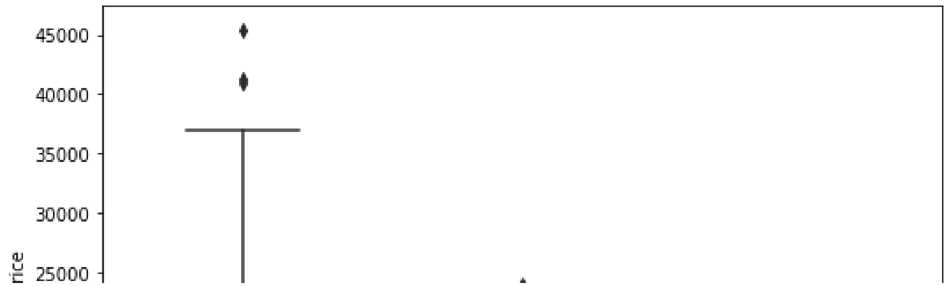
fuelsystem vs Price

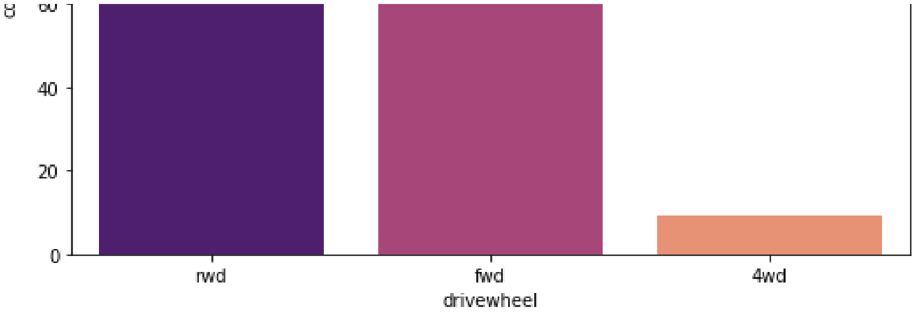


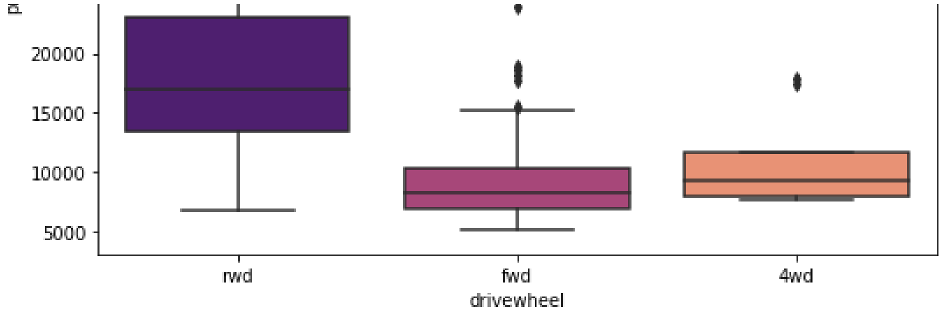
drivewheel Histogram



drivewheel vs Price







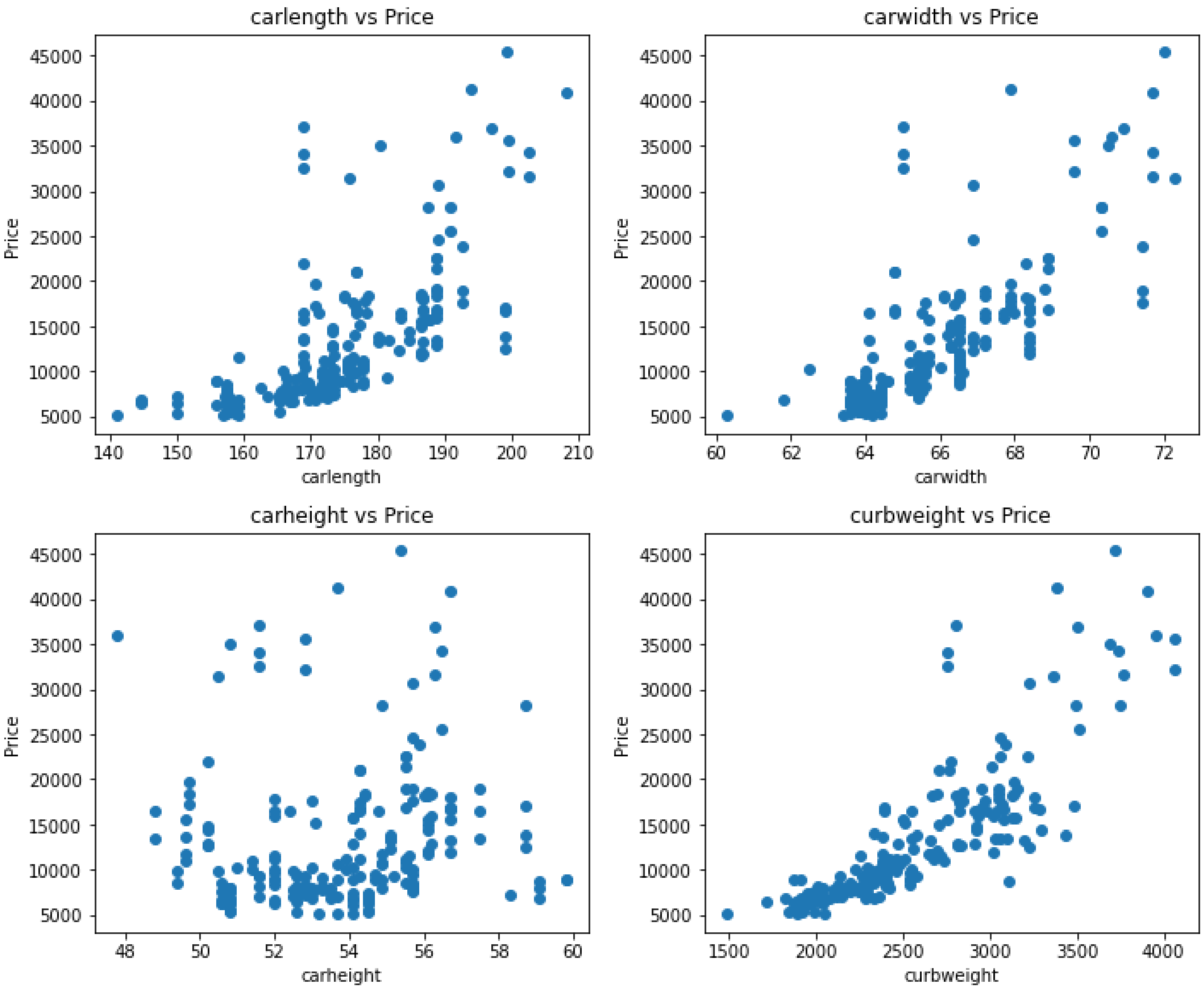
Inference :

1. Very few datapoints for `enginelocation` categories to make an inference.
2. Most common number of cylinders are `four` , `six` and `five` . Though `eight` cylinders have the highest price range.
3. `mpfi` and `2bbl` are most common type of fuel systems. `mpfi` and `idi` having the highest price range. But there are few data for other categories to derive any meaningful inference
4. A very significant difference in drivewheel category. Most high ranged cars seeme to prefer `rwd` drivewheel.

Step 2 : Visualising numerical data

In [50]:

```
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)

plt.tight_layout()
```



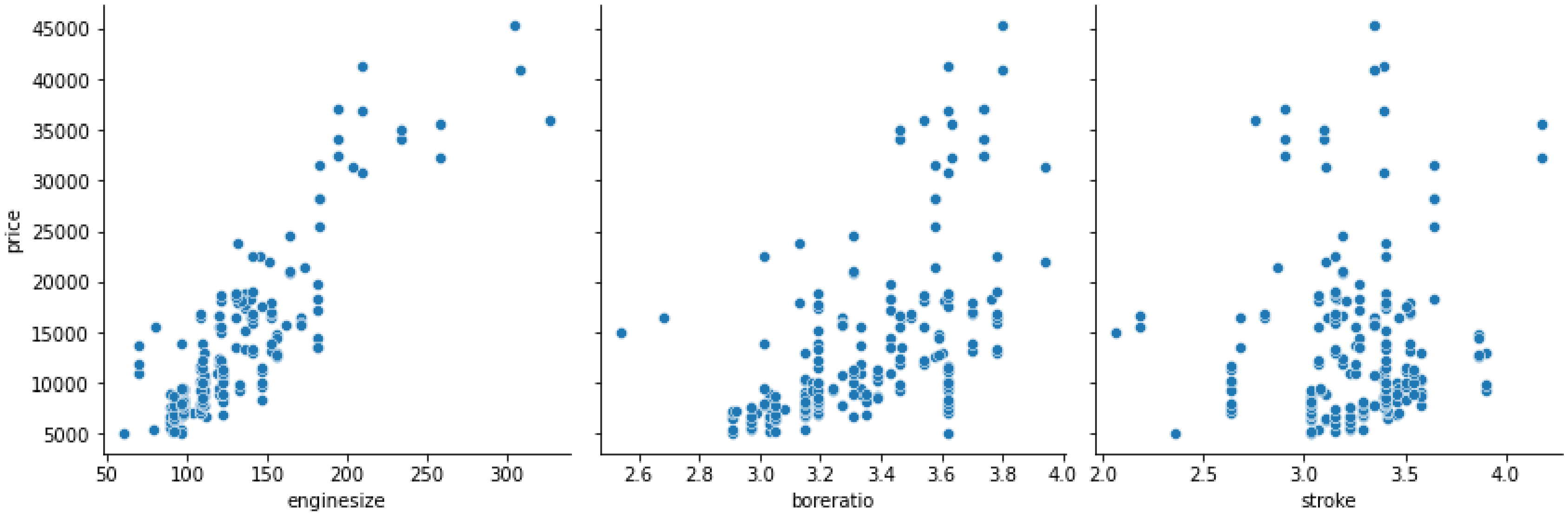
Inference :

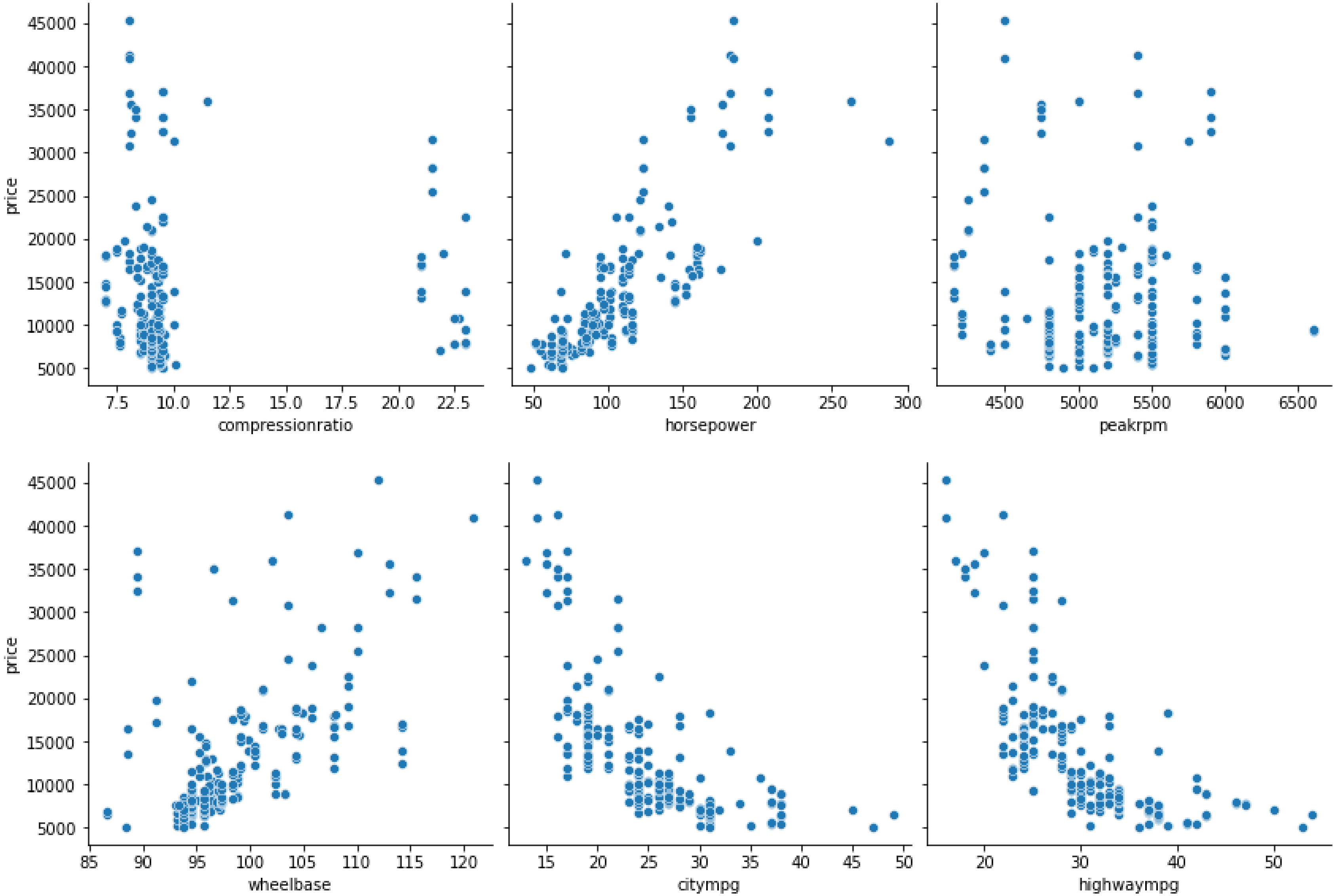
1. `carwidth` , `carlength` and `curbweight` seems to have a poitive correlation with `price` .
2. `carheight` doesn't show any significant trend with price.

In [51]:
```
#Outliers also can be treated in this section
```

c. Do you want to exclude some variables from the model based on this analysis? What other actions will you take? (2 marks)

In [52]:
```
def pp(x,y,z):
    sns.pairplot(cars, x_vars=[x,y,z], y_vars='price',size=4, aspect=1, kind='scatter')
    plt.show()

pp('enginesize', 'boreratio', 'stroke')
pp('compressionratio', 'horsepower', 'peakrpm')
pp('wheelbase', 'citympg', 'highwaympg')
```





Inference :

1. `enginesize` , `boreratio` , `horsepower` , `wheelbase` - seem to have a significant positive correlation with price.
2. `citympg` , `highwaympg` - seem to have a significant negative correlation with price.

In [ ]:
```
np.corrcoef(cars['carlength'], cars['carwidth'])[0, 1]
```

Out[ ]: 0.841118268481846

features engineering

In [ ]:
```
#Fuel economy
cars['fueleconomy'] = (0.55 * cars['citympg']) + (0.45 * cars['highwaympg'])
```

In [ ]:
```
#Binning the Car Companies based on avg prices of each Company.
cars['price'] = cars['price'].astype('int')
temp = cars.copy()
table = temp.groupby(['CompanyName'])['price'].mean()
temp = temp.merge(table.reset_index(), how='left',on='CompanyName')
bins = [0,10000,20000,40000]
cars_bin=['Budget','Medium','Highend']
cars['carsrange'] = pd.cut(temp['price_y'],bins,right=False,labels=cars_bin)
cars.head()
```

Out[ ]:

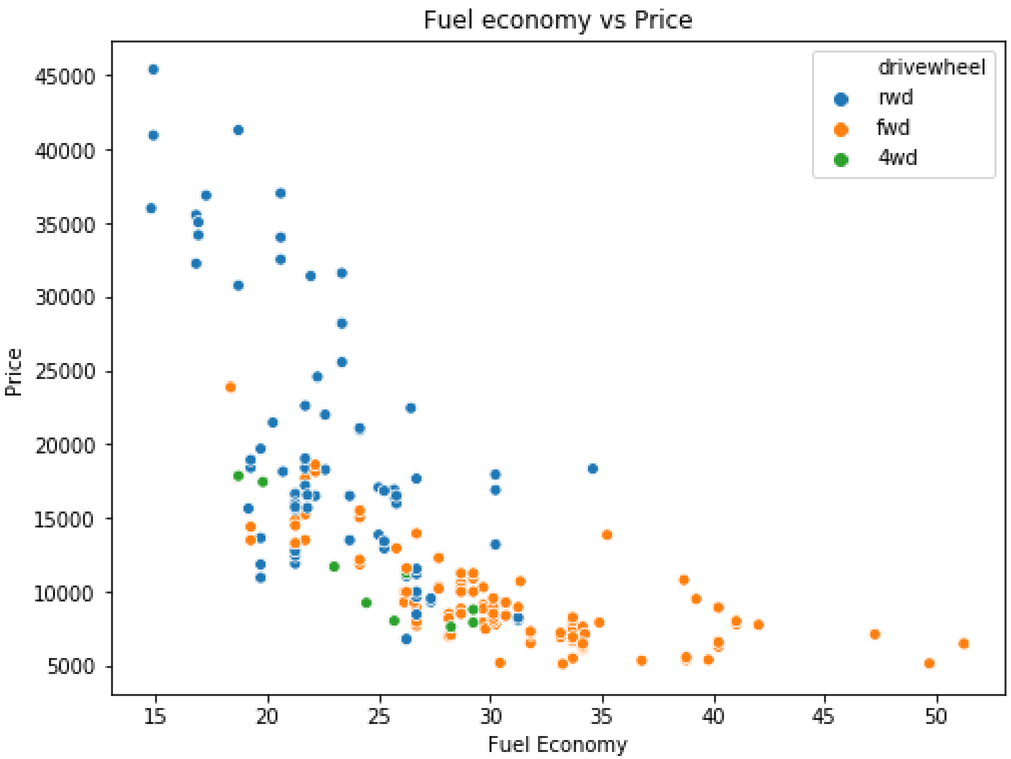
	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	boreratio	stroke	compr
0	1	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	
1	2	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.47	2.68	
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	2.68	3.47	
3	4	2	audi	gas	std	four	sedan	fwd	front	99.8	...	3.19	3.40	
4	5	2	audi	gas	std	four	sedan	4wd	front	99.4	...	3.19	3.40	

5 rows × 28 columns

Bivariate Analysis

In [ ]:
```
plt.figure(figsize=(8,6))

plt.title('Fuel economy vs Price')
sns.scatterplot(x=cars['fueleconomy'],y=cars['price'],hue=cars['drivewheel'])
plt.xlabel('Fuel Economy')
plt.ylabel('Price')
```

```
plt.show()
plt.tight_layout()
```



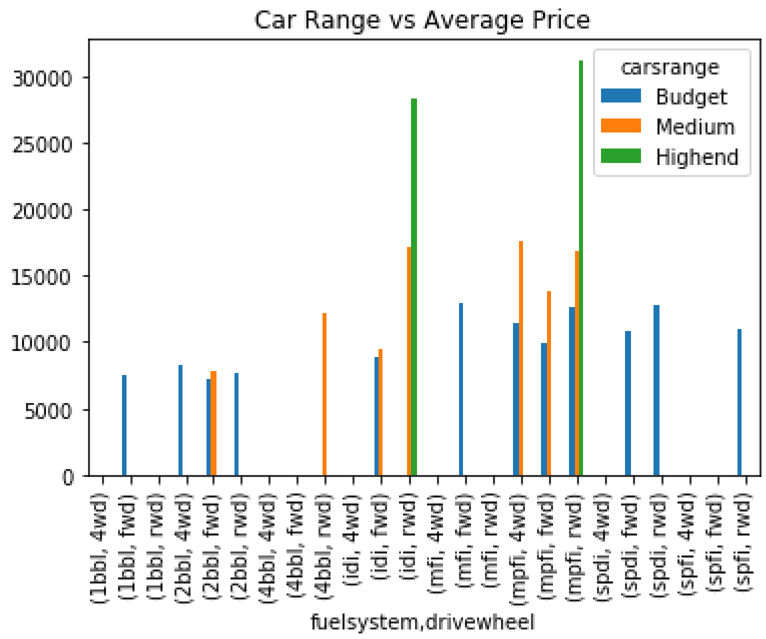
<Figure size 432x288 with 0 Axes>

Inference :

1. `fueleconomy` has an obvios `negative correlation` with price and is significant.

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

df = pd.DataFrame(cars.groupby(['fuelsystem','drivewheel','carsrange'])['price'].mean().unstack(fill_value=0))
df.plot.bar()
plt.title('Car Range vs Average Price')
plt.show()
```

<Figure size 1800x432 with 0 Axes>



Inference :

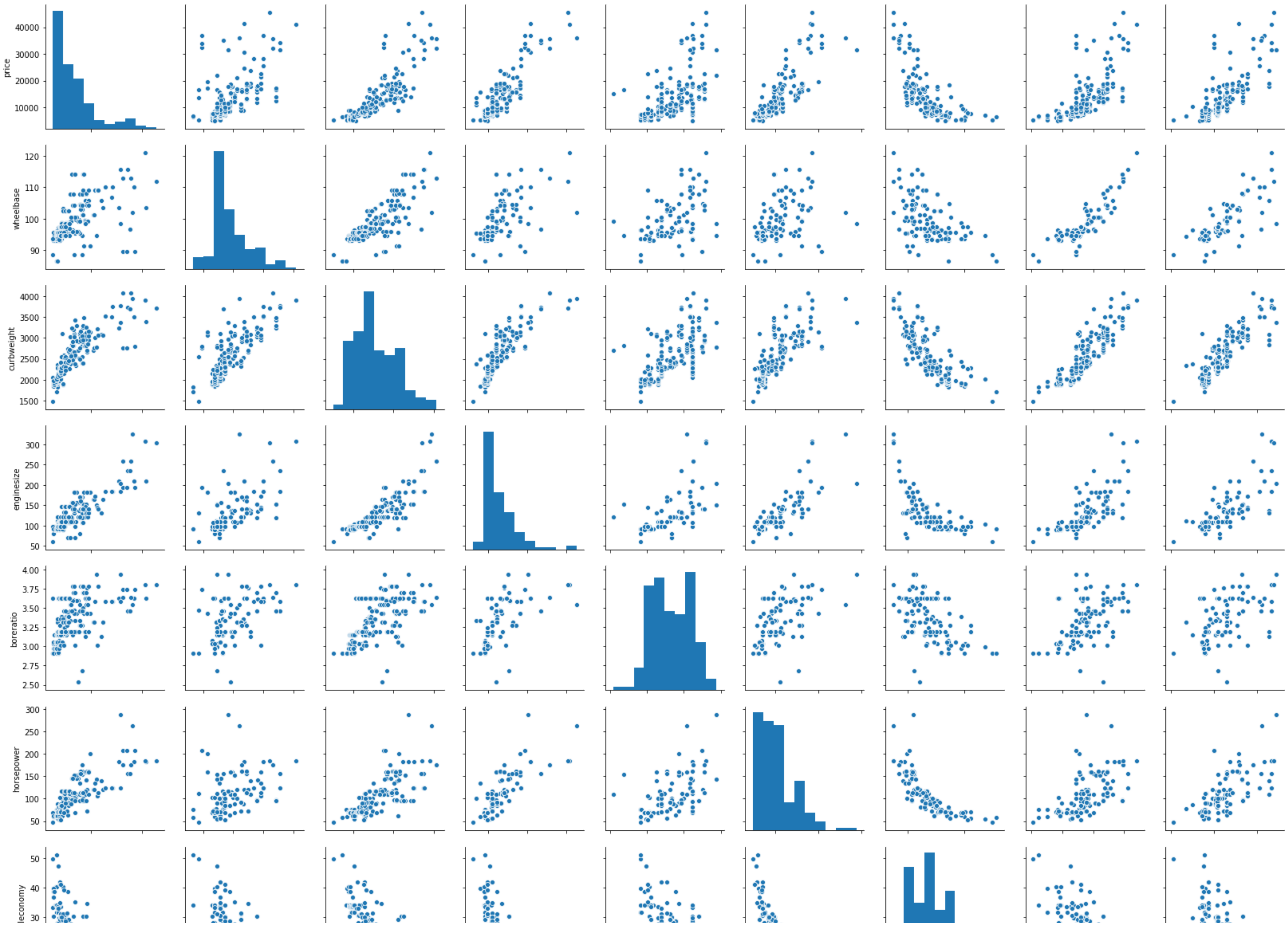
1. High ranged cars prefer `rwd` drivewheel with `idi` or `mpfi` fuelsystem.

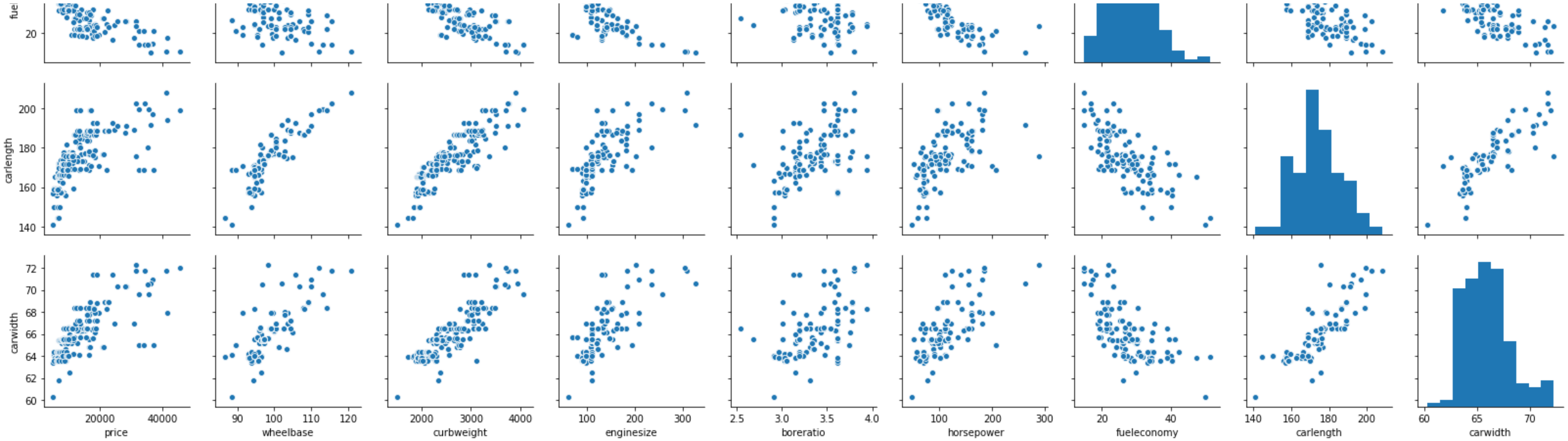
In [ ]:
```
cars_lr = cars[['price', 'fueltype', 'aspiration','carbody', 'drivewheel','wheelbase',
                  'curbweight', 'enginetype', 'cylindernumber', 'enginesize', 'boreratio','horsepower',
                    'fueleconomy', 'carlength','carwidth', 'carsrange']]
cars_lr.head()
```

Out[ ]:

	price	fueltype	aspiration	carbody	drivewheel	wheelbase	curbweight	enginetype	cylindernumber	enginesize	boreratio	horsepower	fuelecono
0	13495	gas	std	convertible	rwd	88.6	2548	dohc	four	130	3.47	111	23
1	16500	gas	std	convertible	rwd	88.6	2548	dohc	four	130	3.47	111	23
2	16500	gas	std	hatchback	rwd	94.5	2823	ohcv	six	152	2.68	154	22
3	13950	gas	std	sedan	fwd	99.8	2337	ohc	four	109	3.19	102	26
4	17450	gas	std	sedan	4wd	99.4	2824	ohc	five	136	3.19	115	19

In [ ]:
```
sns.pairplot(cars_lr)
plt.show()
```





In [ ]:
```
# Defining the map function
def dummies(x,df):
    temp = pd.get_dummies(df[x], drop_first = True)
    df = pd.concat([df, temp], axis = 1)
    df.drop([x], axis = 1, inplace = True)
    return df
# Applying the function to the cars_lr

cars_lr = dummies('fueltype',cars_lr)
cars_lr = dummies('aspiration',cars_lr)
cars_lr = dummies('carbody',cars_lr)
cars_lr = dummies('drivewheel',cars_lr)
cars_lr = dummies('enginetype',cars_lr)
cars_lr = dummies('cylindernumber',cars_lr)
cars_lr = dummies('carsrange',cars_lr)
```

In [ ]:
```
cars_lr.head()
```

Out[ ]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	rotor	five	four	six	three	twe
0	13495	88.6	2548	130	3.47	111	23.70	168.8	64.1	1	...	0	0	0	1	0	0	
1	16500	88.6	2548	130	3.47	111	23.70	168.8	64.1	1	...	0	0	0	1	0	0	
2	16500	94.5	2823	152	2.68	154	22.15	171.2	65.5	1	...	1	0	0	0	1	0	
3	13950	99.8	2337	109	3.19	102	26.70	176.6	66.2	1	...	0	0	0	1	0	0	

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	rotor	five	four	six	three	twe
4	17450	99.4	2824	136	3.19	115	19.80	176.6	66.4	1	...	0	0	1	0	0	0	

5 rows × 31 columns

In [ ]:
```
cars_lr.shape
```

Out[ ]: (205, 31)

d. Split dataset into train and test (70:30). Are both train and test representative of the overall data? How would you ascertain this statistically? (3 marks)

In [ ]:
```
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_state = 100)
```

In [ ]:
```
stats.ttest_ind(df_train.iloc[:,1:], df_test.iloc[:,1:])
#All the pvalues> 0.05
```

Out[ ]:
```
Ttest_indResult(statistic=array([-0.84259363, -0.66959192, -1.01793949, -1.80371206, -0.85980387,
        0.4894942 , -1.11542253, -0.68781997,  0.48536645,  0.07485128,
       -2.03683   ,  0.0544803 ,  0.31367245,  0.72274638,  0.3971903 ,
       -0.94651433,  0.65754088,  0.40577066, -1.78317972,  0.89459542,
       -0.04240254,  1.32919754,  0.89270227, -1.79580298,  1.06593877,
        0.65754088,  0.65754088,  1.32919754, -0.30370793,  0.12168329]), pvalue=array([0.40044739, 0.50387885, 0.30991836, 0.0727
5905, 0.39091141,
       0.62502037, 0.26598809, 0.49235132, 0.62793983, 0.94040675,
       0.04296531, 0.95660608, 0.75409191, 0.47066756, 0.69164443,
       0.34501162, 0.51157807, 0.68533848, 0.0760512 , 0.37206285,
       0.96621948, 0.18527472, 0.37307364, 0.07401301, 0.28771745,
       0.51157807, 0.51157807, 0.18527472, 0.76166147, 0.90327021]))
```

In [ ]:
```
stats.ttest_ind(df_train.iloc[:,0], df_test.iloc[:,0])
#All the pvalues> 0.05
```

Out[ ]: Ttest_indResult(statistic=-0.5988626606978698, pvalue=0.5499321581575893)

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

```
scaler = MinMaxScaler()
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepower','fueleconomy','carlength','carwidth','price']
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

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

Out[ ]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	rotor	five	four	six	three
122	0.068818	0.244828	0.272692	0.139623	0.230159	0.083333	0.530864	0.426016	0.291667	1	...	0	0	0	1	0	0
125	0.466890	0.272414	0.500388	0.339623	1.000000	0.395833	0.213992	0.452033	0.666667	1	...	0	0	0	1	0	0
166	0.122110	0.272414	0.314973	0.139623	0.444444	0.266667	0.344307	0.448780	0.308333	1	...	0	0	0	1	0	0
1	0.314446	0.068966	0.411171	0.260377	0.626984	0.262500	0.244170	0.450407	0.316667	1	...	0	0	0	1	0	0
199	0.382131	0.610345	0.647401	0.260377	0.746032	0.475000	0.122085	0.775610	0.575000	1	...	0	0	0	1	0	0

5 rows × 31 columns

In [ ]:
```
df_test.head()
```

Out[ ]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	rotor	five	four	six	three	tw
160	7738	95.7	2094	98	3.19	70	42.05	166.3	64.4	1	...	0	0	0	1	0	0	
186	8495	97.3	2275	109	3.19	85	30.15	171.7	65.5	1	...	0	0	0	1	0	0	
59	8845	98.8	2385	122	3.39	84	28.70	177.8	66.5	1	...	0	0	0	1	0	0	
165	9298	94.5	2265	98	3.24	112	27.35	168.7	64.0	1	...	0	0	0	1	0	0	
140	7603	93.3	2240	108	3.62	73	28.25	157.3	63.8	1	...	0	0	0	1	0	0	

5 rows × 31 columns

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

Out[ ]:

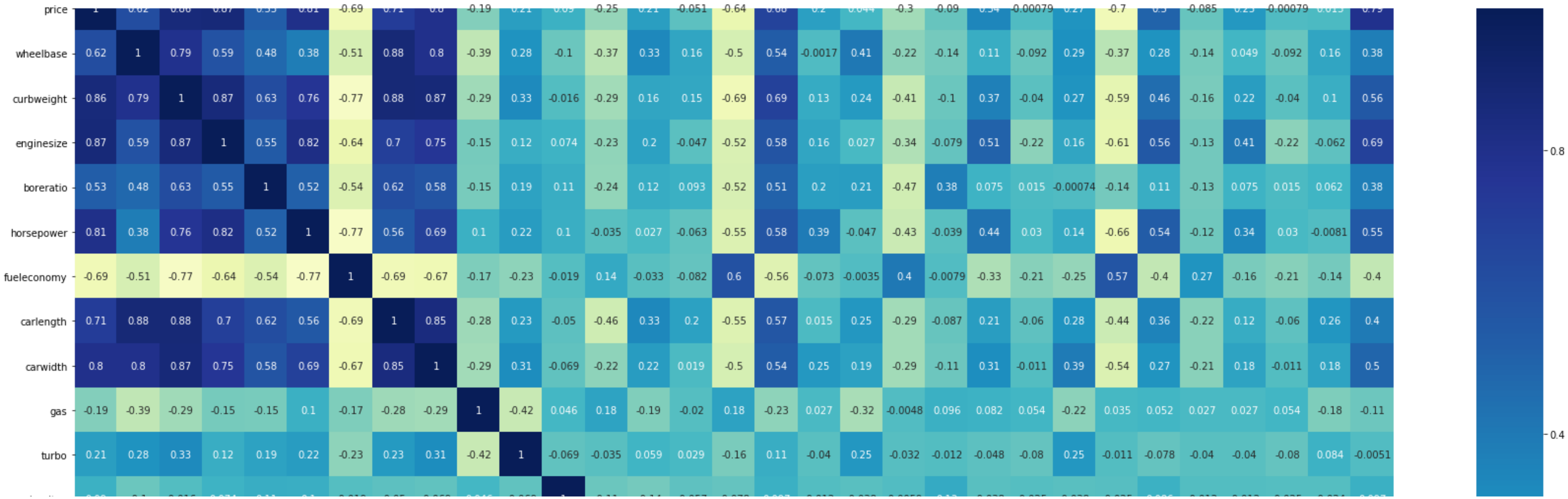
	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	ro

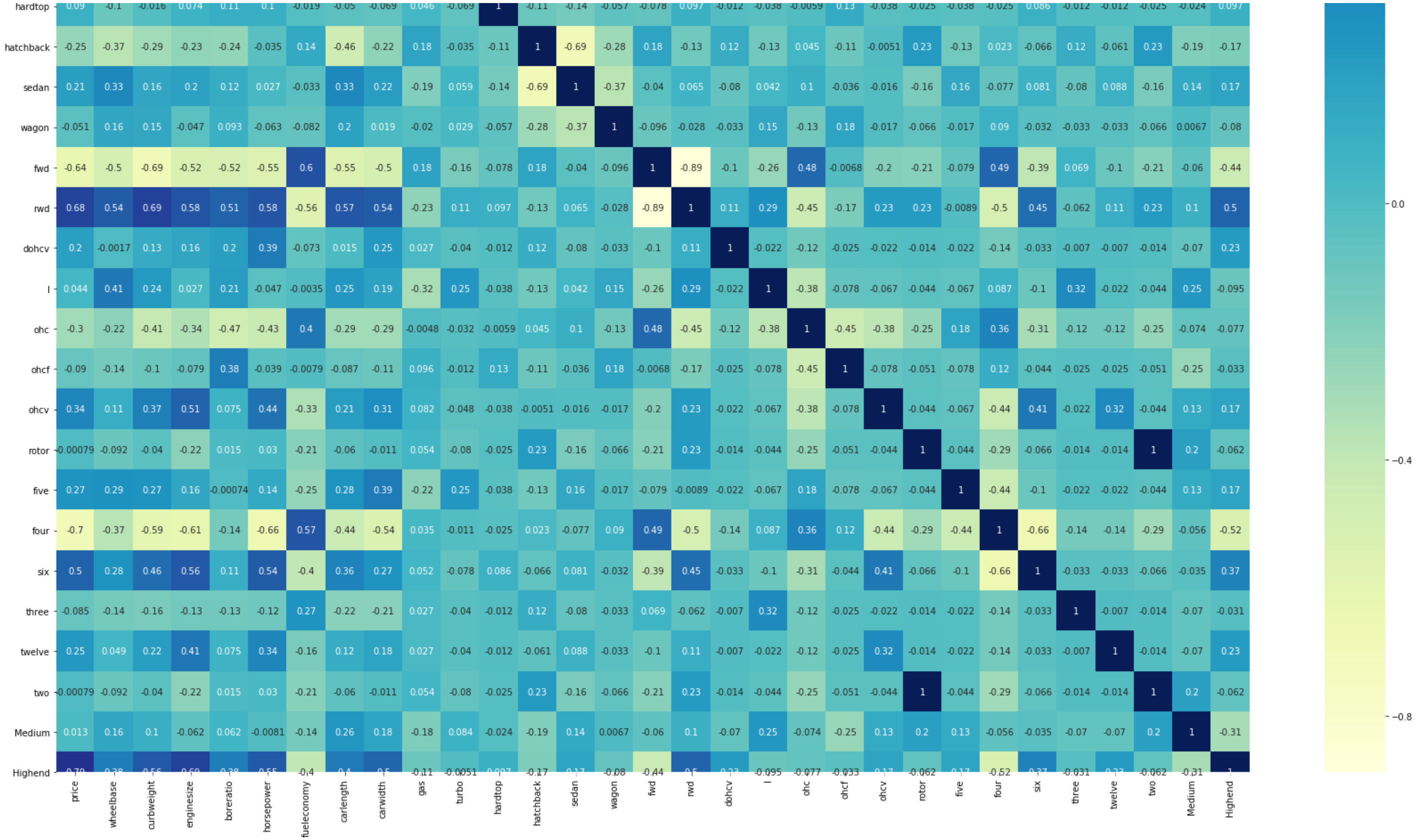
	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	ro
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	...	143.000000	143.000(
mean	0.219309	0.411141	0.407878	0.241351	0.497946	0.227302	0.358265	0.525476	0.461655	0.909091	...	0.062937	0.0279
std	0.215682	0.205581	0.211269	0.154619	0.207140	0.165511	0.185980	0.204848	0.184517	0.288490	...	0.243703	0.1654
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000(
25%	0.067298	0.272414	0.245539	0.135849	0.305556	0.091667	0.198903	0.399187	0.304167	1.000000	...	0.000000	0.000(
50%	0.140343	0.341379	0.355702	0.184906	0.500000	0.191667	0.344307	0.502439	0.425000	1.000000	...	0.000000	0.000(
75%	0.313479	0.503448	0.559542	0.301887	0.682540	0.283333	0.512346	0.669919	0.550000	1.000000	...	0.000000	0.000(
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000(

8 rows × 31 columns

In [ ]:

```python
#Correlation using heatmap
plt.figure(figsize = (30, 25))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```





Highly correlated variables to price are - `curbweight` , `enginesize` , `horsepower` , `carwidth` and `highend` .

In [ ]:
```
#Dividing data into X and y variables
y_train = df_train.pop('price')
X_train = df_train
```

In [ ]:
```
from scipy import stats
```

3. Model Building (20 marks)

a. Fit a base model and observe the overall R- Squared, RMSE and MAPE values of the model. Please comment on whether it is good or not. (5 marks)

b. Check for multi-collinearity and treat the same. (3 marks)

c. How would you improve the model? Write clearly the changes that you will make before re-fitting the model. Fit the final model. (6 marks)

d. Write down a business interpretation/explanation of the model – which variables are affecting the target the most and explain the relationship. Feel free to use charts or graphs to explain. (4 marks)

e. What changes from the base model had the most effect on model performance? (2 marks)

a. Fit a base model and observe the overall R- Squared, RMSE and MAPE values of the model. Please comment on whether it is good or not. (5 marks)

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

In [ ]:
```
lm = LinearRegression()
lm.fit(X_train,y_train)
rfe = RFE(lm, 10)
rfe = rfe.fit(X_train, y_train)
```

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

Out[ ]:
```
Index(['curbweight', 'horsepower', 'fueleconomy', 'carwidth', 'hatchback',
       'sedan', 'wagon', 'dohcv', 'twelve', 'Highend'],
      dtype='object')
```

Building model using statsmodel, for the detailed statistics

In [ ]:
```
X_train_rfe = X_train[X_train.columns[rfe.support_]]
X_train_rfe.head()
```

Out[ ]:

	curbweight	horsepower	fueleconomy	carwidth	hatchback	sedan	wagon	dohcv	twelve	Highend
122	0.272692	0.083333	0.530864	0.291667	0	1	0	0	0	0

	curbweight	horsepower	fueleconomy	carwidth	hatchback	sedan	wagon	dohcv	twelve	Highend
125	0.500388	0.395833	0.213992	0.666667	1	0	0	0	0	1
166	0.314973	0.266667	0.344307	0.308333	1	0	0	0	0	0
1	0.411171	0.262500	0.244170	0.316667	0	0	0	0	0	0
199	0.647401	0.475000	0.122085	0.575000	0	0	1	0	0	0

In [ ]:

```
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)
```

MODEL 1

In [ ]:

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.929
Model:                            OLS   Adj. R-squared:                  0.923
Method:                 Least Squares   F-statistic:                     172.1
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           1.29e-70
Time:                        13:58:33   Log-Likelihood:                 205.85
No. Observations:                 143   AIC:                            -389.7
Df Residuals:                     132   BIC:                            -357.1
Df Model:                          10
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const          -0.0947      0.042     -2.243      0.027      -0.178      -0.011
curbweight      0.2657      0.069      3.870      0.000       0.130       0.402
```

```
horsepower      0.4499      0.074      6.099      0.000       0.304       0.596
fueleconomy     0.0933      0.052      1.792      0.075      -0.010       0.196
carwidth        0.2609      0.062      4.216      0.000       0.138       0.383
hatchback      -0.0929      0.025     -3.707      0.000      -0.143      -0.043
sedan          -0.0704      0.025     -2.833      0.005      -0.120      -0.021
wagon          -0.0997      0.028     -3.565      0.001      -0.155      -0.044
dohcv          -0.2676      0.079     -3.391      0.001      -0.424      -0.112
twelve         -0.1192      0.067     -1.769      0.079      -0.253       0.014
Highend         0.2586      0.020     12.929      0.000       0.219       0.298
==============================================================================
Omnibus:                       43.093   Durbin-Watson:                   1.867
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              130.648
Skew:                           1.128   Prob(JB):                     4.27e-29
Kurtosis:                       7.103   Cond. No.                         32.0
==============================================================================

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

p-vale of `twelve` seems to be higher than the significance value of 0.05, hence dropping it as it is insignificant in presence of other variables.

Here overall R- Squared

- R-squared: 0.929 Adj. R-squared: 0.923

In [ ]:
```python
X_train_new = X_train_rfe.drop(["twelve"], axis = 1)
```

In [ ]:
```python
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.927
Model:                            OLS   Adj. R-squared:                  0.922
Method:                 Least Squares   F-statistic:                     187.9
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           4.25e-71
Time:                        13:58:33   Log-Likelihood:                 204.17
No. Observations:                 143   AIC:                            -388.3
Df Residuals:                     133   BIC:                            -358.7
Df Model:                           9
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const          -0.0764      0.041     -1.851      0.066      -0.158       0.005
curbweight      0.2756      0.069      3.995      0.000       0.139       0.412
horsepower      0.3997      0.069      5.824      0.000       0.264       0.535
```

```
fueleconomy     0.0736      0.051      1.435      0.154      -0.028       0.175
carwidth        0.2580      0.062      4.137      0.000       0.135       0.381
hatchback      -0.0951      0.025     -3.766      0.000      -0.145      -0.045
sedan          -0.0744      0.025     -2.983      0.003      -0.124      -0.025
wagon          -0.1050      0.028     -3.744      0.000      -0.160      -0.050
dohcv          -0.2319      0.077     -3.015      0.003      -0.384      -0.080
Highend         0.2565      0.020     12.743      0.000       0.217       0.296
==============================================================================
Omnibus:                       48.027   Durbin-Watson:                   1.880
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              159.802
Skew:                           1.231   Prob(JB):                     1.99e-35
Kurtosis:                       7.556   Cond. No.                         29.6
==============================================================================

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

In [ ]:

```
X_train_new = X_train_new.drop(["fueleconomy"], axis = 1)  #pvalue>0.05
```

MODEL 3

In [ ]:

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.926
Model:                            OLS   Adj. R-squared:                  0.922
Method:                 Least Squares   F-statistic:                     209.5
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           7.85e-72
Time:                        13:58:34   Log-Likelihood:                 203.07
No. Observations:                 143   AIC:                            -388.1
Df Residuals:                     134   BIC:                            -361.5
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0305      0.026     -1.165      0.246      -0.082       0.021
curbweight     0.2593      0.068      3.796      0.000       0.124       0.394
horsepower     0.3469      0.058      5.964      0.000       0.232       0.462
carwidth       0.2488      0.062      3.995      0.000       0.126       0.372
hatchback     -0.0922      0.025     -3.650      0.000      -0.142      -0.042
sedan         -0.0711      0.025     -2.850      0.005      -0.120      -0.022
wagon         -0.1047      0.028     -3.721      0.000      -0.160      -0.049
dohcv         -0.1968      0.073     -2.689      0.008      -0.342      -0.052
Highend        0.2610      0.020     13.083      0.000       0.222       0.301
```

```
==============================================================================
Omnibus:                       48.637   Durbin-Watson:                   1.909
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              161.444
Skew:                           1.250   Prob(JB):                     8.77e-36
Kurtosis:                       7.566   Cond. No.                         27.2
==============================================================================

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

variables are significant?

List of significant variables :

```
- Car Range
- Engine Type
- Fuel type
- Car Body
- Aspiration
- Cylinder Number
- Drivewheel
- Curbweight
- Car Length
- Car width
- Engine Size
- Boreratio
- Horse Power
- Wheel base
- Fuel Economy
```

b. Check for multi-collinearity and treat the same. (3 marks)

In [ ]:

```
#Calculating the Variance Inflation Factor
checkVIF(X_train_new)
```

Out[ ]:

	Features	VIF
0	const	26.90
1	curbweight	8.10
5	sedan	6.07

	Features	VIF
4	hatchback	5.63
3	carwidth	5.14
2	horsepower	3.61
6	wagon	3.58
8	Highend	1.63
7	dohcv	1.46

dropping `curbweight` because of high VIF value. (shows that curbweight has high multicollinearity.)

In [ ]:
```
X_train_new = X_train_new.drop(["curbweight"], axis = 1)
```

c. How would you improve the model? Write clearly the changes that you will make before re-fitting the model. Fit the final model. (6 marks)

In [ ]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.918
Model:                            OLS   Adj. R-squared:                  0.914
Method:                 Least Squares   F-statistic:                     215.9
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           4.70e-70
Time:                        13:58:37   Log-Likelihood:                 195.77
No. Observations:                 143   AIC:                            -375.5
Df Residuals:                     135   BIC:                            -351.8
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0319      0.027     -1.161      0.248      -0.086       0.022
horsepower     0.4690      0.051      9.228      0.000       0.368       0.569
carwidth       0.4269      0.043      9.944      0.000       0.342       0.512
hatchback     -0.1044      0.026     -3.976      0.000      -0.156      -0.052
sedan         -0.0756      0.026     -2.896      0.004      -0.127      -0.024
wagon         -0.0865      0.029     -2.974      0.003      -0.144      -0.029
dohcv         -0.3106      0.070     -4.435      0.000      -0.449      -0.172
Highend        0.2772      0.020     13.559      0.000       0.237       0.318
```

```
==============================================================================
Omnibus:                       43.937   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              127.746
Skew:                           1.171   Prob(JB):                     1.82e-28
Kurtosis:                       6.995   Cond. No.                         18.0
==============================================================================

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

In [ ]:
```
checkVIF(X_train_new)
```

Out[ ]:

	Features	VIF
0	const	26.89
4	sedan	6.06
3	hatchback	5.54
5	wagon	3.47
1	horsepower	2.50
2	carwidth	2.22
7	Highend	1.56
6	dohcv	1.21

dropping `sedan` because of high VIF value.

In [ ]:
```
X_train_new = X_train_new.drop(["sedan"], axis = 1)
```

MODEL

In [ ]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.913
Model:                            OLS   Adj. R-squared:                  0.909
Method:                 Least Squares   F-statistic:                     237.6
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           1.68e-69
Time:                        13:58:38   Log-Likelihood:                 191.46
No. Observations:                 143   AIC:                            -368.9
```

```
Df Residuals:                     136   BIC:                            -348.2
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0934      0.018     -5.219      0.000      -0.129      -0.058
horsepower     0.5001      0.051      9.805      0.000       0.399       0.601
carwidth       0.3963      0.043      9.275      0.000       0.312       0.481
hatchback     -0.0373      0.013     -2.938      0.004      -0.062      -0.012
wagon         -0.0170      0.017     -1.008      0.315      -0.050       0.016
dohcv         -0.3203      0.072     -4.460      0.000      -0.462      -0.178
Highend        0.2808      0.021     13.402      0.000       0.239       0.322
==============================================================================
Omnibus:                       34.143   Durbin-Watson:                   2.024
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               72.788
Skew:                           1.018   Prob(JB):                     1.56e-16
Kurtosis:                       5.841   Cond. No.                         16.4
==============================================================================

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

In [ ]:

```
checkVIF(X_train_new)
```

Out[ ]:

	Features	VIF
0	const	10.82
1	horsepower	2.39
2	carwidth	2.09
6	Highend	1.55
3	hatchback	1.23
5	dohcv	1.21
4	wagon	1.11

dropping `wagon` because of high p-value.

In [ ]:

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

MODEL

In [ ]:
```
X_train_new = build_model(X_train_new,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.912
Model:                            OLS   Adj. R-squared:                  0.909
Method:                 Least Squares   F-statistic:                     284.8
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           1.57e-70
Time:                        13:58:39   Log-Likelihood:                 190.93
No. Observations:                 143   AIC:                            -369.9
Df Residuals:                     137   BIC:                            -352.1
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0970      0.018     -5.530      0.000      -0.132      -0.062
horsepower     0.5013      0.051      9.832      0.000       0.401       0.602
carwidth       0.3952      0.043      9.252      0.000       0.311       0.480
hatchback     -0.0336      0.012     -2.764      0.006      -0.058      -0.010
dohcv         -0.3231      0.072     -4.502      0.000      -0.465      -0.181
Highend        0.2833      0.021     13.615      0.000       0.242       0.324
==============================================================================
Omnibus:                       36.097   Durbin-Watson:                   2.028
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               78.717
Skew:                           1.067   Prob(JB):                     8.07e-18
Kurtosis:                       5.943   Cond. No.                         16.3
==============================================================================

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

In [ ]:
```
checkVIF(X_train_new)
```

Out[ ]:

	Features	VIF
0	const	10.39
1	horsepower	2.39
2	carwidth	2.08
5	Highend	1.53
4	dohcv	1.21

	Features	VIF
3	hatchback	1.13

MODEL

In [ ]:
```
#Dropping dohcv to see the changes in model statistics
X_train_new = X_train_new.drop(["dohcv"], axis = 1)
X_train_new = build_model(X_train_new,y_train)
checkVIF(X_train_new)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.899
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     308.0
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           1.04e-67
Time:                        13:58:40   Log-Likelihood:                 181.06
No. Observations:                 143   AIC:                            -352.1
Df Residuals:                     138   BIC:                            -337.3
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0824      0.018     -4.480      0.000      -0.119      -0.046
horsepower     0.4402      0.052      8.390      0.000       0.336       0.544
carwidth       0.3957      0.046      8.677      0.000       0.306       0.486
hatchback     -0.0414      0.013     -3.219      0.002      -0.067      -0.016
Highend        0.2794      0.022     12.591      0.000       0.236       0.323
==============================================================================
Omnibus:                       29.385   Durbin-Watson:                   1.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               98.010
Skew:                           0.692   Prob(JB):                     5.22e-22
Kurtosis:                       6.812   Cond. No.                         12.9
==============================================================================

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

Out[ ]:

	Features	VIF
0	const	10.04
1	horsepower	2.22

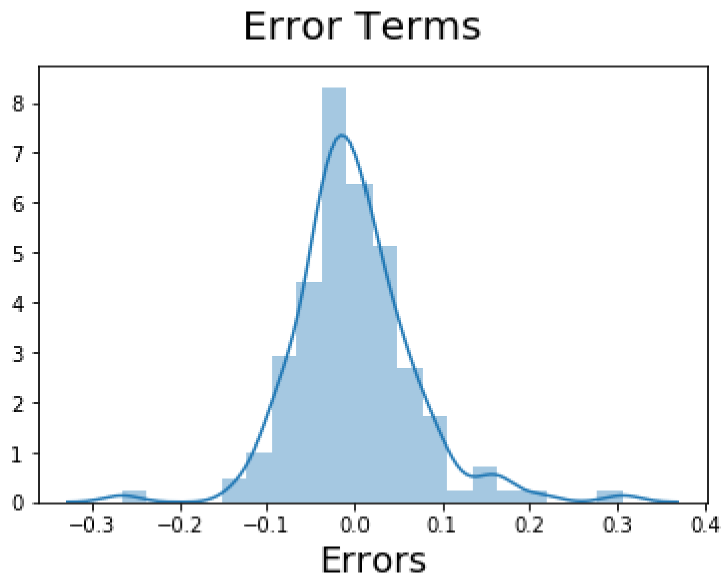
	Features	VIF
2	carwidth	2.08
4	Highend	1.53
3	hatchback	1.10

Residual Analysis of Model

In [ ]:
```
lm = sm.OLS(y_train,X_train_new).fit()
y_train_price = lm.predict(X_train_new)
```

In [ ]:
```
# 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)
```

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



Error terms seem to be approximately normally distributed, so the assumption on the linear modeling seems to be fulfilled.

Prediction and Evaluation

In [ ]:
```
#Scaling the test set
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepower','fueleconomy','carlength','carwidth','price']
df_test[num_vars] = scaler.fit_transform(df_test[num_vars])
```

In [ ]:
```
#Dividing into X and y
y_test = df_test.pop('price')
X_test = df_test
```

In [ ]:
```
# Now let's use our model to make predictions.
X_train_new = X_train_new.drop('const',axis=1)
# Creating X_test_new dataframe by dropping variables from X_test
X_test_new = X_test[X_train_new.columns]

# Adding a constant variable
X_test_new = sm.add_constant(X_test_new)
```

In [ ]:
```
# Making predictions
y_pred = lm.predict(X_test_new)
```

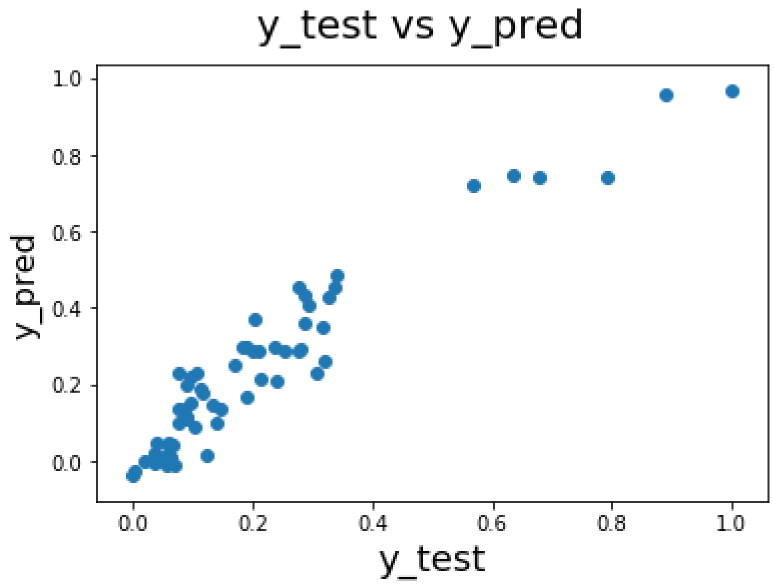
Evaluation of test via comparison of y_pred and y_test

In [ ]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[ ]:
```
0.8614595209022033
```

In [ ]:
```
#EVALUATION OF THE MODEL
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)
```

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



Evaluation of the model using Statistics

In [ ]:
```
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.899
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     308.0
Date:                Wed, 22 Jul 2020   Prob (F-statistic):           1.04e-67
Time:                        13:58:44   Log-Likelihood:                 181.06
No. Observations:                 143   AIC:                            -352.1
Df Residuals:                     138   BIC:                            -337.3
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0824      0.018     -4.480      0.000      -0.119      -0.046
horsepower     0.4402      0.052      8.390      0.000       0.336       0.544
carwidth       0.3957      0.046      8.677      0.000       0.306       0.486
hatchback     -0.0414      0.013     -3.219      0.002      -0.067      -0.016
Highend        0.2794      0.022     12.591      0.000       0.236       0.323
==============================================================================
Omnibus:                       29.385   Durbin-Watson:                   1.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               98.010
```

```
Skew:                           0.692   Prob(JB):                     5.22e-22
Kurtosis:                       6.812   Cond. No.                         12.9
==============================================================================

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

d. Write down a business interpretation/explanation of the model – which variables are affecting the target the most and explain the relationship. Feel free to use charts or graphs to explain. (4 marks)

1. *R-sqaured and Adjusted R-squared (extent of fit)* - 0.899 and 0.896 - `90%` variance explained.
2. *F-stats and Prob(F-stats) (overall model fit)* - 308.0 and 1.04e-67(approx. 0.0) - Model fir is significant and explained `90%` variance is just not by chance.
3. *p-values* - p-values for all the coefficients seem to be less than the significance level of 0.05. - meaning that all the predictors are statistically significant.

In [ ]:

```

```