In [2]:

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

In [3]:

```
import numpy as np
import pandas as pd
```

In [4]:

```
cars = pd.read_csv("machine learning data/CarPrice_Assignment.csv")
cars.head()
```

Out[4]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136

5 rows × 26 columns

In [5]:

```
cars.shape
```

Out[5]:

```
(205, 26)
```

In [6]:

```
cars.info()
```

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

```
peakrpm             205 non-null int64
citympg             205 non-null int64
highwaympg          205 non-null int64
price               205 non-null float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.7+ KB
```

In [7]:

```python
cars.describe()
```

Out[7]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	com
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	

In [8]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(15,8))

plt.subplot(1,2,1)
plt.title('car prize distributioin')
sns.distplot(cars.price)


plt.subplot(1,2,2)
plt.title("car price spread")
sns.boxplot(y=cars.price)
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x226ec1d5080>
```

In [9]:

```python
print(cars.price.describe())
```

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

In [10]:

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

Out[10]:

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

5 rows × 26 columns

In [11]:

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

Out[11]:

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

In [12]:

```
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[12]:

```
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 [13]:

```
cars.duplicated()
```

Out[13]:

```
0       False
1       False
2       False
3       False
4       False
5       False
6       False
7       False
8       False
9       False
10      False
11      False
12      False
13      False
14      False
15      False
16      False
17      False
18      False
19      False
20      False
```

```
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
       ...
175    False
176    False
177    False
178    False
179    False
180    False
181    False
182    False
183    False
184    False
185    False
186    False
187    False
188    False
189    False
190    False
191    False
192    False
193    False
194    False
195    False
196    False
197    False
198    False
199    False
200    False
201    False
202    False
203    False
204    False
Length: 205, dtype: bool
```

In [14]:

```
cars.columns
```

Out[14]:

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

In [ ]:

```

```

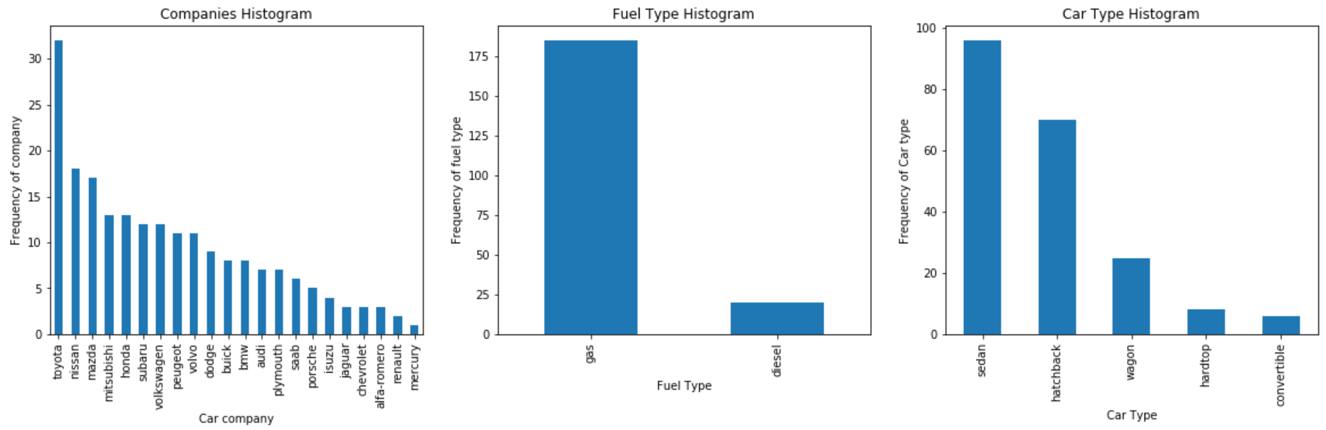
visualising categorical data

In [15]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,3,1)

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')
```

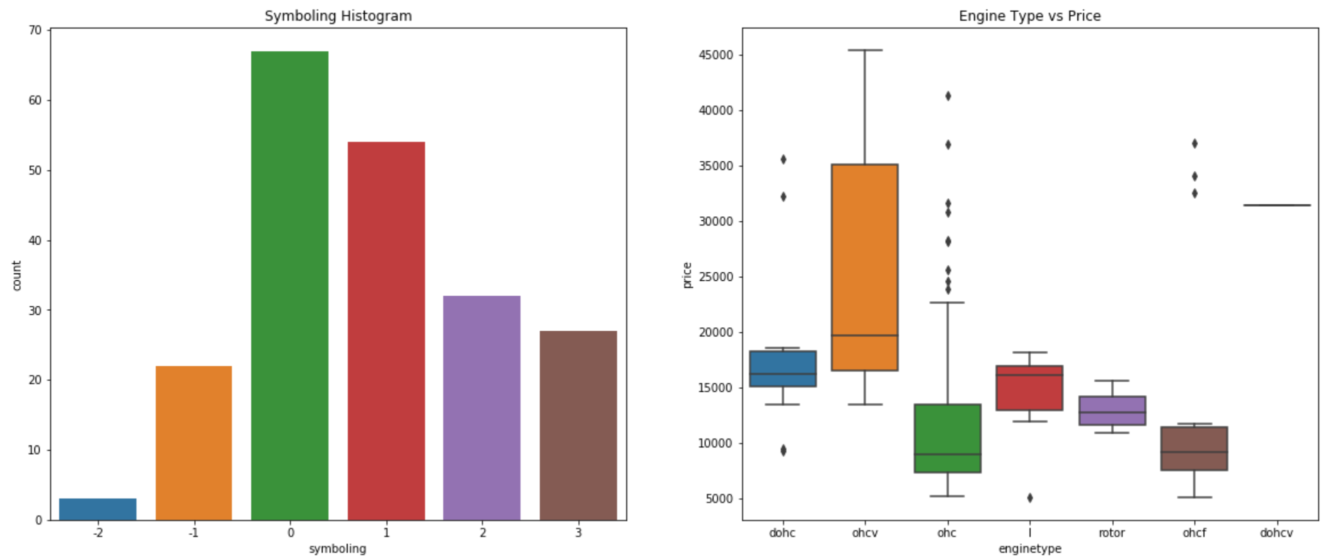
```
plt.subplot(1,3,2)
plt1 = cars.fueltype.value_counts().plot('bar')
plt.title('Fuel Type Histogram')
plt1.set(xlabel = 'Fuel Type', ylabel='Frequency of fuel type')

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()
```

```
C:\Users\Pujitha\Anaconda3\lib\site-packages\matplotlib\figure.py:98:
MatplotlibDeprecationWarning:
Adding an axes using the same arguments as a previous axes currently reuses the earlier instance.
In a future version, a new instance will always be created and returned.  Meanwhile, this warning
can be suppressed, and the future behavior ensured, by passing a unique label to each axes instanc
e.
  "Adding an axes using the same arguments as a previous axes "
```



In [16]:

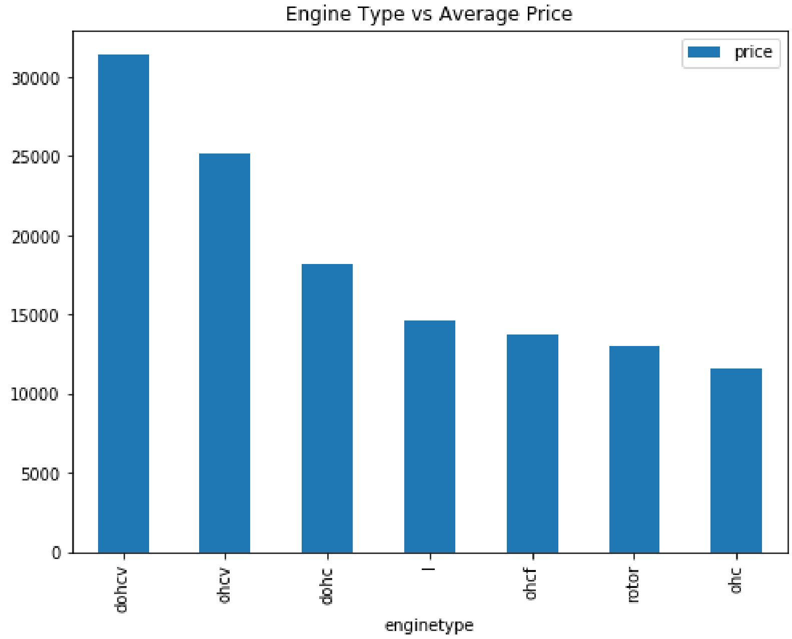
```
plt.figure(figsize=(20,8))

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

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

plt.show()
```



In [17]:

In [17]:

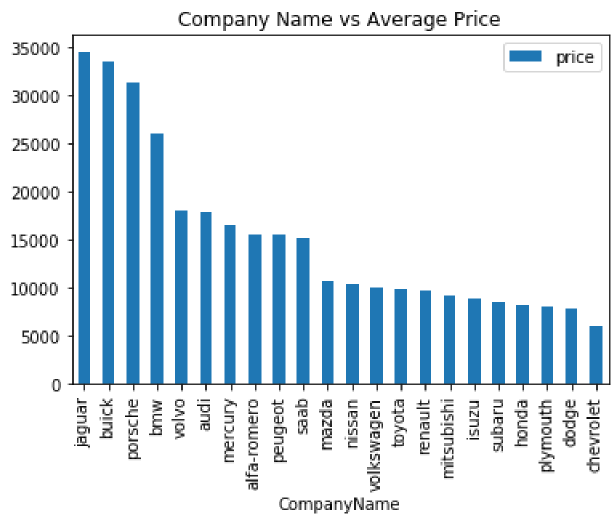
```
df = pd.DataFrame(cars.groupby(['enginetype'])['price'].mean().sort_values(ascending = False))
df.plot.bar(figsize=(8,6))
plt.title('Engine Type vs Average Price')
plt.show()
```



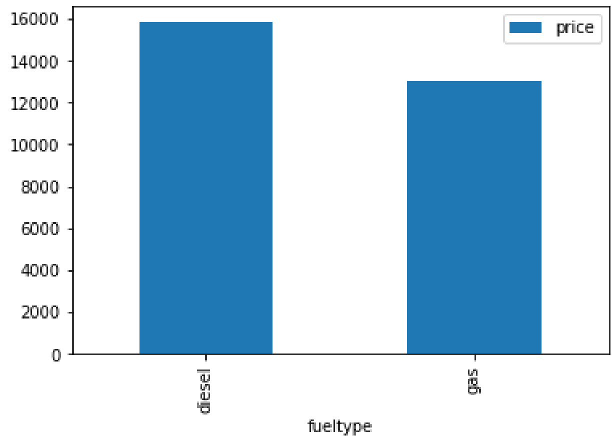
In [18]:

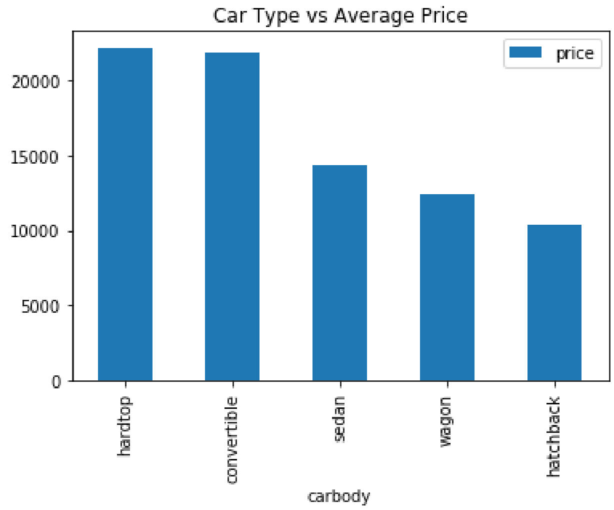
```
plt.figure(figsize=(20,6))

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 1440x432 with 0 Axes>
```



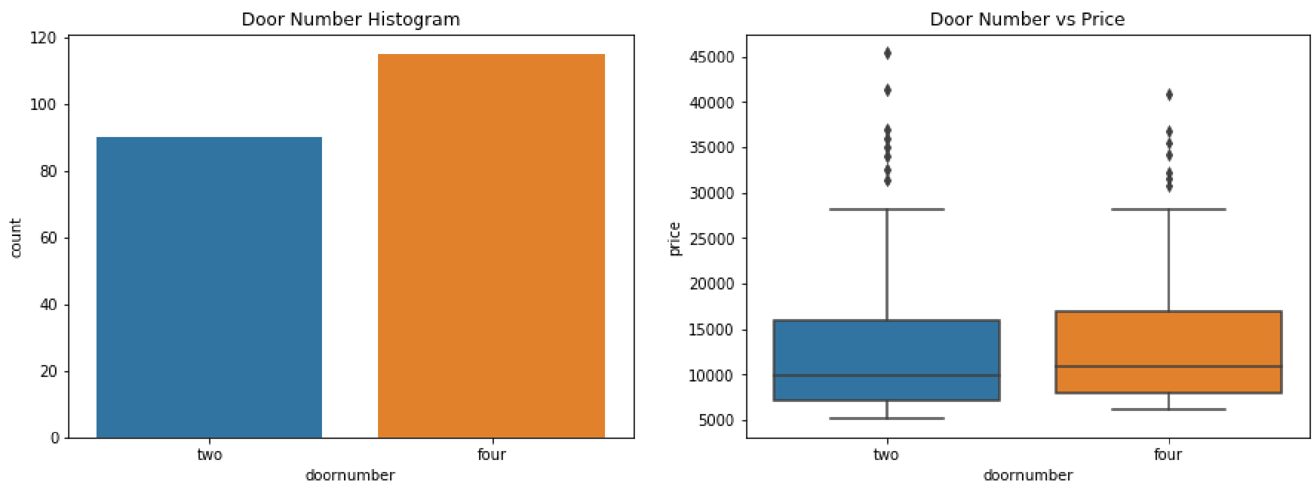
Fuel Type vs Average Price





In [19]:

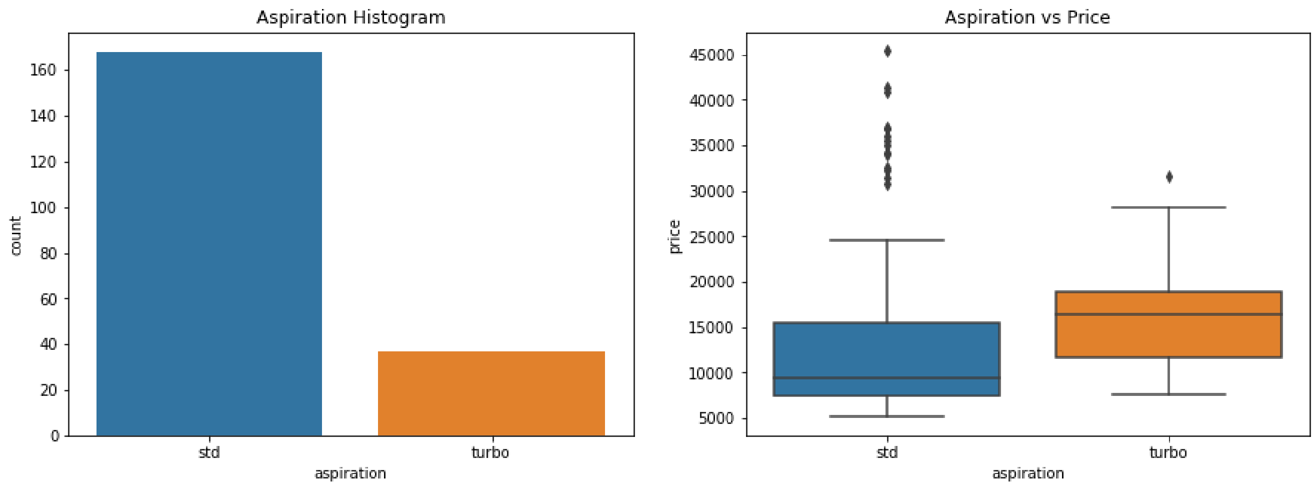
```
plt.figure(figsize=(15,5))

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

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

plt.show()
```



In [20]:

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

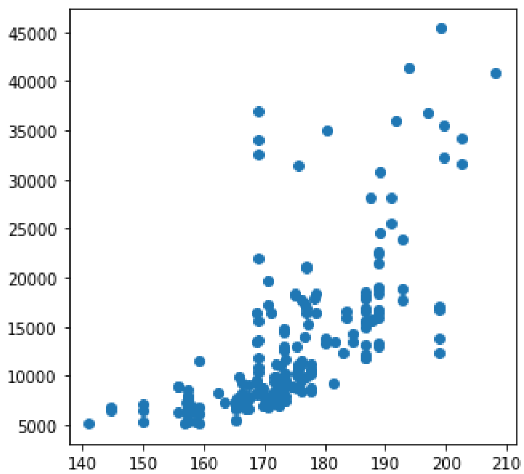
```
plt.subplot(1,2,1)
plt.title('Aspiration Histogram')
sns.countplot(cars.aspiration)

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

plt.show()
```

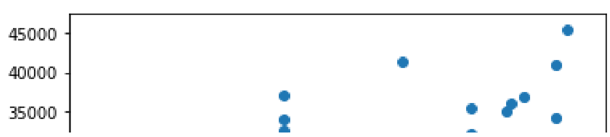


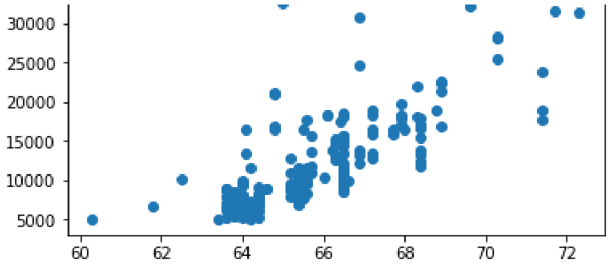
visuvalising numarical data

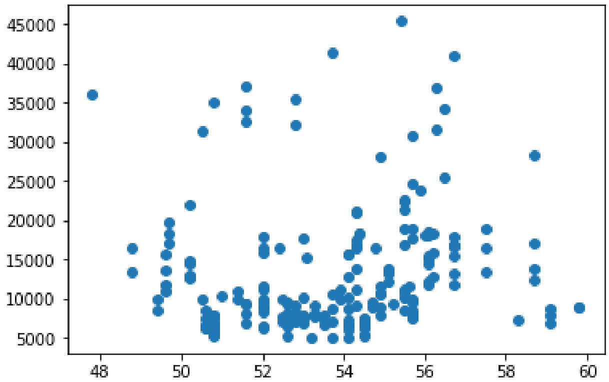
In [21]:

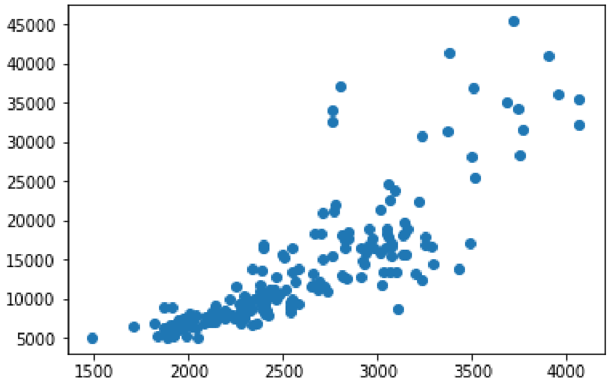
```
plt.figure(figsize=[5,5])
plt.scatter(cars.carlength,cars.price)
plt.show()

#plt.figure(figsize=[5,5])
plt.scatter(cars.carwidth,cars.price)
plt.show()

#plt.figure(figsize=[5,5])
plt.scatter(cars.carheight,cars.price)
plt.show()

#plt.figure(figsize=[5,5])
plt.scatter(cars.curbweight,cars.price)
plt.show()
```







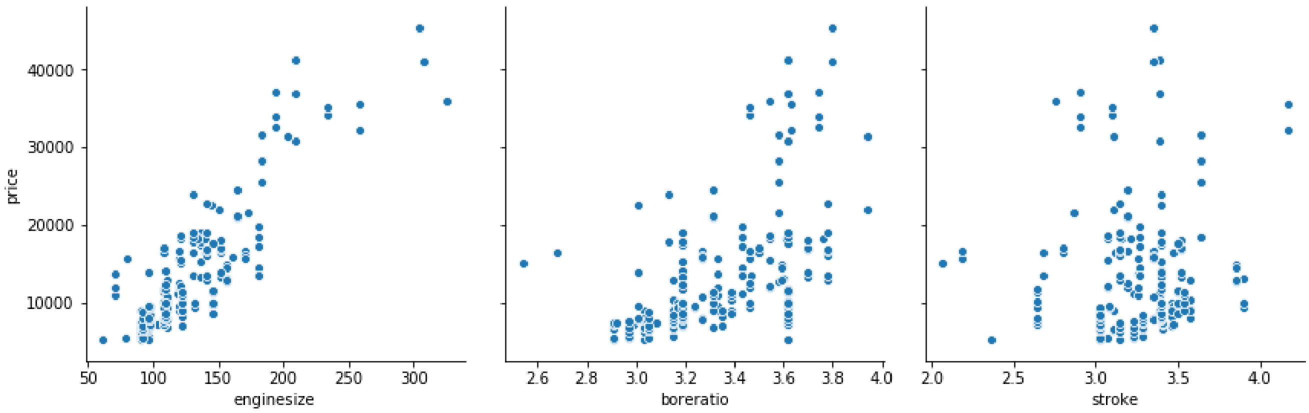




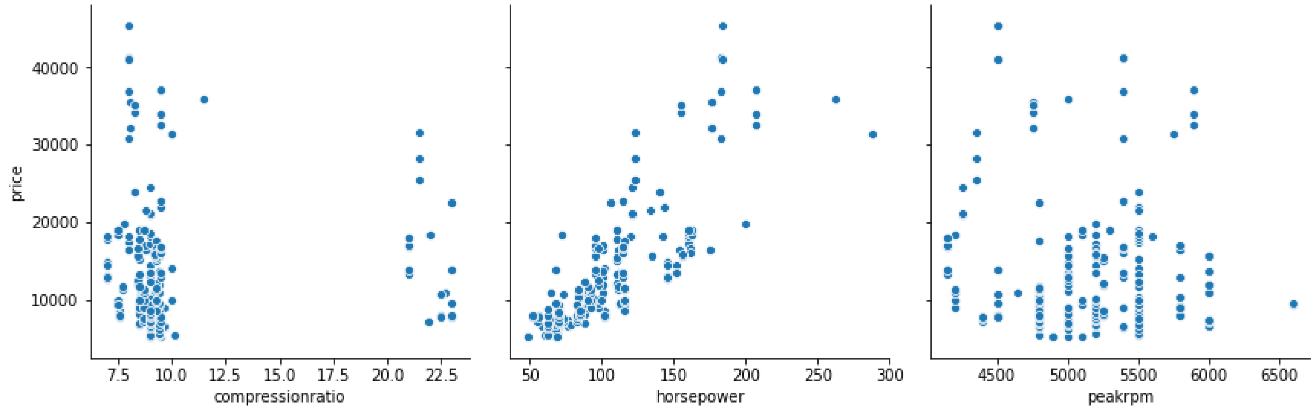
In [22]:

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

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

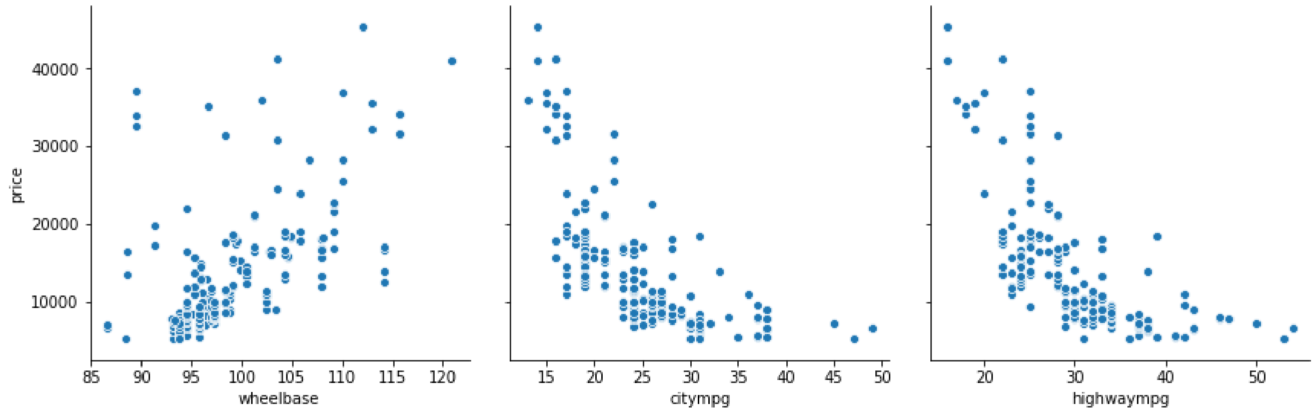
```
C:\Users\Pujitha\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065: UserWarning: The `size` par
ameter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```



```
C:\Users\Pujitha\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065: UserWarning: The `size` par
ameter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```




```
C:\Users\Pujitha\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065: UserWarning: The `size` par
ameter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```



In [23]:

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

Out[23]:

```
0.841118268481846
```

In [24]:

```
cars['fueleconomy'] = (0.55 * cars['citympg']) + (0.45 * cars['highwaympg'])
```

In [25]:

```
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[25]:

	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	borerat
0	1	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.4
1	2	3	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	3.4

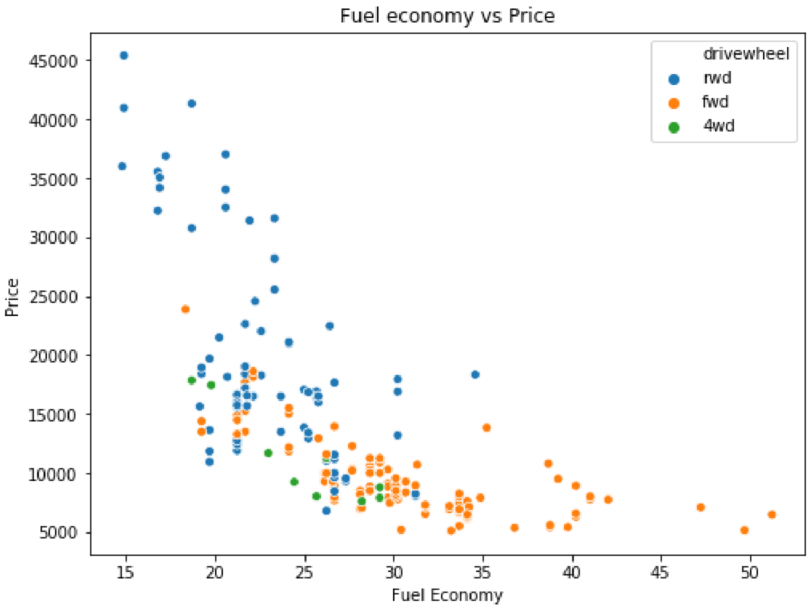
2	car_ID	symboling	CompanyName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	borerat
2	3	1	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	2.6
3	4	2	audi	gas	std	four	sedan	fwd	front	99.8	...	3.1
4	5	2	audi	gas	std	four	sedan	4wd	front	99.4	...	3.1

5 rows × 28 columns

bivariate analysis

In [26]:

```
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()
```

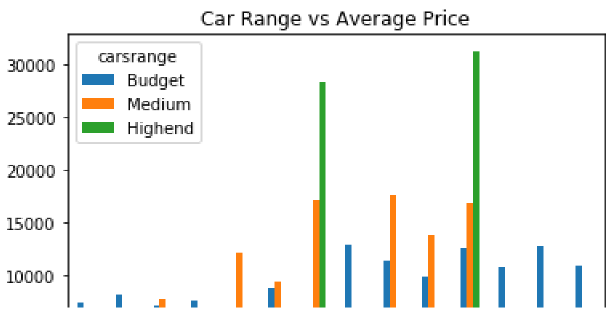


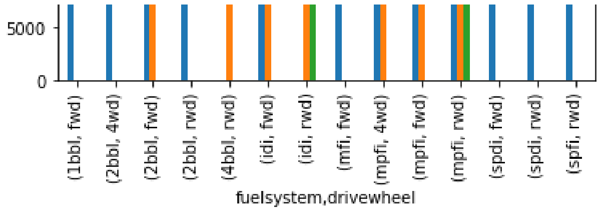
<Figure size 432x288 with 0 Axes>

In [27]:

```
plt.figure(figsize=(25, 6))

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

<Figure size 1800x432 with 0 Axes>





In [33]:

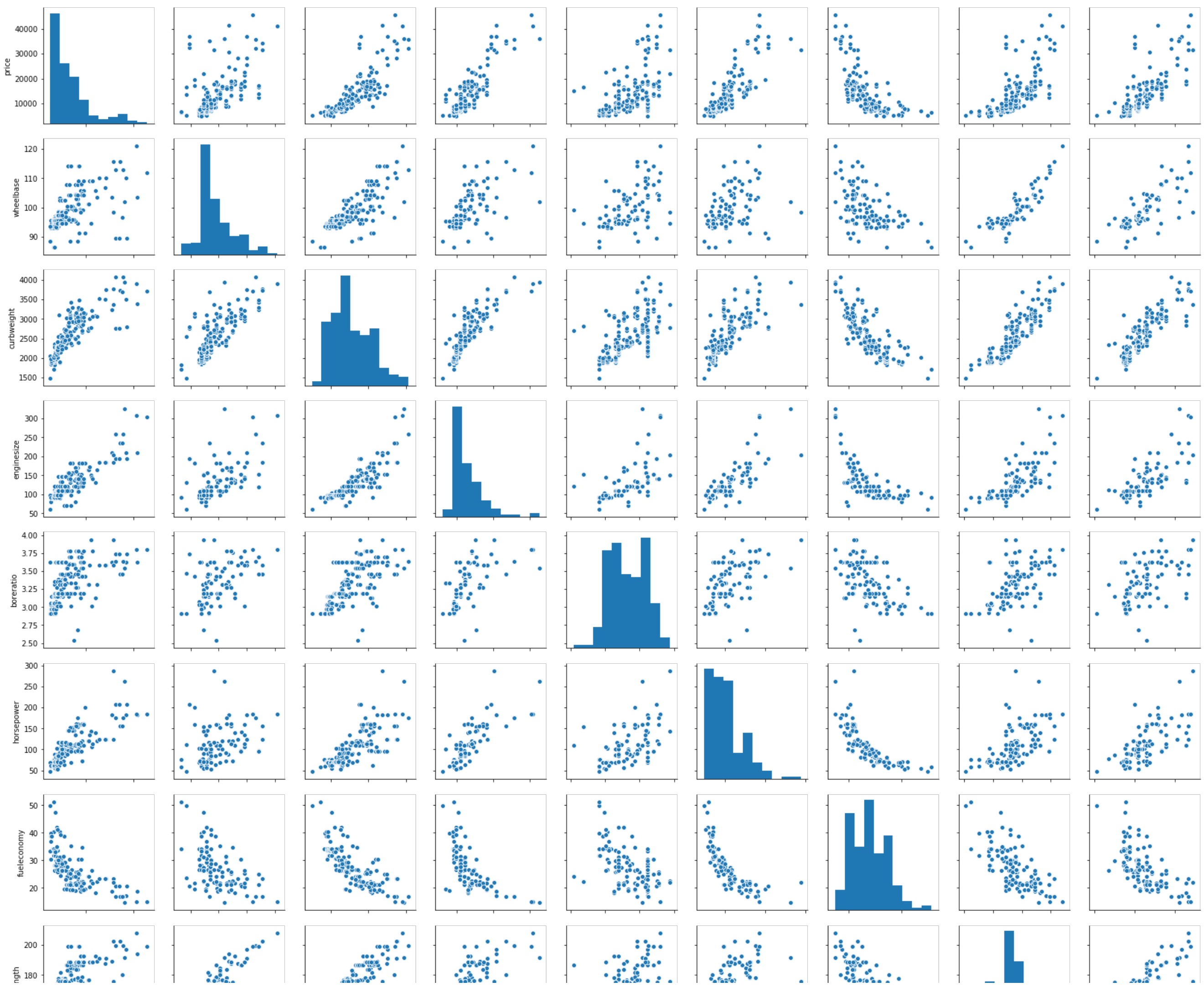
```
cars_lr = cars[['price', 'fueltype', 'aspiration','carbody', 'drivewheel','wheelbase',
                  'curbweight', 'enginetype', 'cylindernumber', 'enginesize',
'boreratio','horsepower',
                    'fueleconomy', 'carlength','carwidth', 'carsrange']]
cars_lr.head()
```

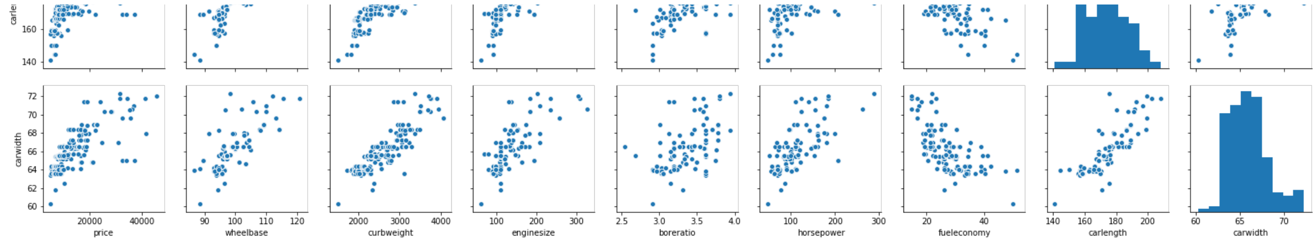
Out[33]:

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

In [34]:

```
sns.pairplot(cars_lr)
plt.show()
```





dummiy variable

In [35]:

```
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 [36]:

```
cars_lr.head()
```

Out[36]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	rotor	five	fou
0	13495	88.6	2548	130	3.47	111	23.70	168.8	64.1	1	...	0	0	0	
1	16500	88.6	2548	130	3.47	111	23.70	168.8	64.1	1	...	0	0	0	
2	16500	94.5	2823	152	2.68	154	22.15	171.2	65.5	1	...	1	0	0	
3	13950	99.8	2337	109	3.19	102	26.70	176.6	66.2	1	...	0	0	0	
4	17450	99.4	2824	136	3.19	115	19.80	176.6	66.4	1	...	0	0	1	

5 rows × 31 columns

In [38]:

```
cars_lr.shape
```

Out[38]:

```
(205, 31)
```

..

---

test and train part

In [39]:

```
from sklearn.model_selection import train_test_split
```

In [40]:

```
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 [41]:

```python
from sklearn.preprocessing import MinMaxScaler

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

```
C:\Users\Pujitha\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:334:
DataConversionWarning: Data with input dtype int32, int64, float64 were all converted to float64 b
y MinMaxScaler.
  return self.partial_fit(X, y)
```

In [42]:

```python
df_train.head()
```

Out[42]:

	price	wheelbase	curbweight	enginesize	boreratio	horsepower	fueleconomy	carlength	carwidth	gas	...	ohcv	rotor	five
122	0.068818	0.244828	0.272692	0.139623	0.230159	0.083333	0.530864	0.426016	0.291667	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	(
166	0.122110	0.272414	0.314973	0.139623	0.444444	0.266667	0.344307	0.448780	0.308333	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	(
199	0.382131	0.610345	0.647401	0.260377	0.746032	0.475000	0.122085	0.775610	0.575000	1	...	0	0	(

5 rows × 31 columns

In [43]:

```python
df_train.describe()
```

Out[43]:

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

8 rows × 31 columns

In [44]:

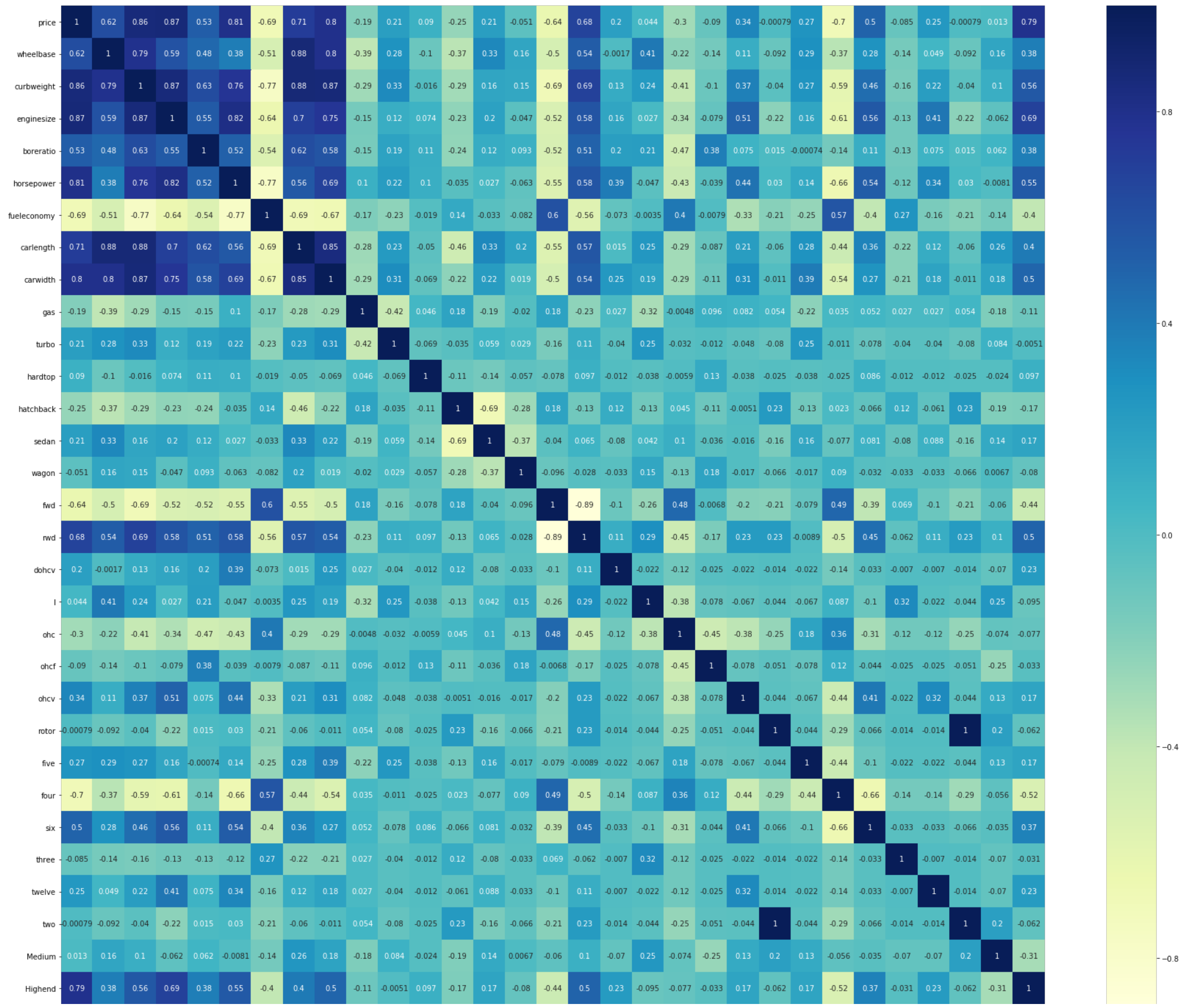
```python
df_train.info()
```

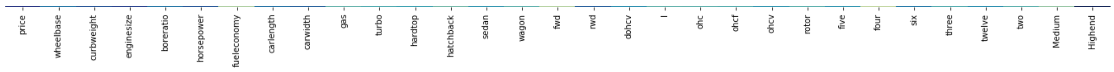
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 143 entries, 122 to 8
Data columns (total 31 columns):
price          143 non-null float64
wheelbase      143 non-null float64
curbweight     143 non-null float64
enginesize     143 non-null float64
boreratio      143 non-null float64
horsepower     143 non-null float64
fueleconomy    143 non-null float64
```

```
carlength       143 non-null float64
carwidth        143 non-null float64
gas             143 non-null uint8
turbo           143 non-null uint8
hardtop         143 non-null uint8
hatchback       143 non-null uint8
sedan           143 non-null uint8
wagon           143 non-null uint8
fwd             143 non-null uint8
rwd             143 non-null uint8
dohcv           143 non-null uint8
l               143 non-null uint8
ohc             143 non-null uint8
ohcf            143 non-null uint8
ohcv            143 non-null uint8
rotor           143 non-null uint8
five            143 non-null uint8
four            143 non-null uint8
six             143 non-null uint8
three           143 non-null uint8
twelve          143 non-null uint8
two             143 non-null uint8
Medium          143 non-null uint8
Highend         143 non-null uint8
dtypes: float64(9), uint8(22)
memory usage: 14.2 KB
```

In [47]:

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





In [48]:

```
#dividing the data into x and y variable
y_train = df_train.pop('price')
X_train = df_train
```

model building part

In [49]:

```
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 [50]:

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

In [51]:

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

Out[51]:

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

In [52]:

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

Out[52]:

	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
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 [57]:

```
X_train_ref=sm.add_constant(X_train['twelve'])
lr = sm.OLS(y_train, X_train_ref).fit()
```

In [58]:

```
lr.params
```

Out[58]:

```
const     0.214845
twelve    0.638319
dtype: float64
```

In [60]:

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.061
Model:                            OLS   Adj. R-squared:                  0.055
Method:                 Least Squares   F-statistic:                     9.200
Date:                Fri, 24 Apr 2020   Prob (F-statistic):            0.00288
Time:                        16:28:06   Log-Likelihood:                 21.468
No. Observations:                 143   AIC:                            -38.94
Df Residuals:                     141   BIC:                            -33.01
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2148      0.018     12.208      0.000       0.180       0.250
twelve         0.6383      0.210      3.033      0.003       0.222       1.054
==============================================================================
Omnibus:                       51.201   Durbin-Watson:                   1.917
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              100.297
Skew:                           1.647   Prob(JB):                     1.66e-22
Kurtosis:                       5.447   Cond. No.                         12.0
==============================================================================

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

In [61]:

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

In [62]:

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

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

In [63]:

```
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:                Fri, 24 Apr 2020   Prob (F-statistic):           1.29e-70
Time:                        16:32:06   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.
```

In [64]:

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

In [65]:

```
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:                Fri, 24 Apr 2020   Prob (F-statistic):           4.25e-71
Time:                        16:34:21   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 [66]:

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

In [67]:

```
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:                Fri, 24 Apr 2020   Prob (F-statistic):           7.85e-72
Time:                        16:35:07   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.
```

In [68]:

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

Out[68]:

	Features	VIF
0	const	26.90
1	curbweight	8.10
5	sedan	6.07
4	hatchback	5.63
3	carwidth	5.14
2	horsepower	3.61
6	wagon	3.58
8	Highend	1.63
7	dohcv	1.46

In [69]:

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

In [70]:

```
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:                Fri, 24 Apr 2020   Prob (F-statistic):           4.70e-70
Time:                        16:36:31   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 [71]:

```
checkVIF(X_train_new)
```

Out[71]:

	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

In [72]:

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

In [73]:

```
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:                Fri, 24 Apr 2020   Prob (F-statistic):           1.68e-69
Time:                        16:37:22   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 [74]:

```
checkVIF(X_train_new)
```

Out[74]:

	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

In [75]:

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

In [76]:

```
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:                Fri, 24 Apr 2020   Prob (F-statistic):           1.57e-70
Time:                        16:38:36   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 [85]:

```
checkVIF(X_train_new)
```

Out[85]:

	Features	VIF

	Features	VIF
0	const	10.04
1	horsepower	2.22
2	carwidth	2.08
4	Highend	1.53
3	hatchback	1.10

residual analysis od a model*

In [88]:

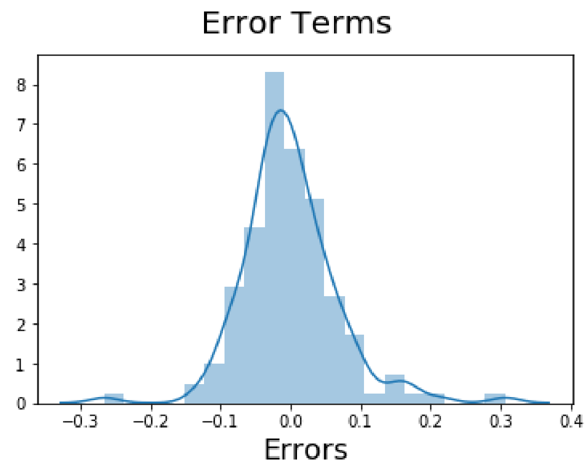
```
lm = sm.OLS(y_train,X_train_new).fit()
y_train_price = lm.predict(X_train_new)
```

In [89]:

```
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[89]:

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



prediction and evaluation part

In [90]:

```
num_vars = ['wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepower','fueleconomy','carle
ngth','carwidth','price']
df_test[num_vars] = scaler.fit_transform(df_test[num_vars])
```

```
C:\Users\Pujitha\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:334:
DataConversionWarning: Data with input dtype int32, int64, float64 were all converted to float64 b
y MinMaxScaler.
  return self.partial_fit(X, y)
```

In [91]:

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

In [92]:

```
X_train_new = X_train_new.drop('const',axis=1)
```

In [93]:

```
X_test_new = X_test[X_train_new.columns]
```

In [94]:

```
#adding  CONSTANT variable
X_test_new = sm.add_constant(X_test_new)
```

In [95]:

```
# Making predictions
y_pred = lm.predict(X_test_new)
```

In [96]:

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

Out[96]:

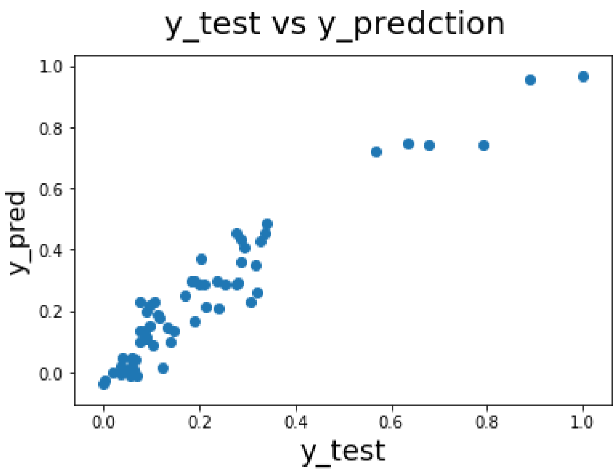
```
0.8614595209022033
```

In [97]:

```
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_predction', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)
```

Out[97]:

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



In [98]:

```
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:                Fri, 24 Apr 2020   Prob (F-statistic):           1.04e-67
Time:                        16:50:50   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.
```

at final all the variables are in permissible limit and the model looks to be stable and good

In [ ]: