In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv(r'C:\Users\aishw\OneDrive\Desktop\ML Assignment 2\Automobile data
```

In [3]:
```
data.head()
```

Out[3]:

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

5 rows × 25 columns

In [4]:
```
data.tail()
```

Out[4]:

	symboling	make	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wh
200	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	
201	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	
202	-1	volvo 244dl	gas	std	four	sedan	rwd	front	
203	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	
204	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	

5 rows × 25 columns

In [5]:
```
data.shape
```

Out[5]:
```
(205, 25)
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
```

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

In [7]:
```
data.describe()
```

Out[7]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	borera
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940

In [8]:
```
data.nunique()
```

Out[8]:
```
symboling           6
make              147
fueltype            2
aspiration          2
doornumber          2
carbody             5
drivewheel          3
enginelocation      2
wheelbase          53
carlength          75
carwidth           44
carheight          49
curbweight        171
enginetype          7
```

```
cylindernumber        7
enginesize           44
fuelsystem            8
boreratio            38
stroke               37
compressionratio     32
horsepower           59
peakrpm              23
citympg              29
highwaympg           30
price               189
dtype: int64
```

In [9]:
```
print(data['fueltype'].value_counts())
print(data['aspiration'].value_counts())
print(data['doornumber'].value_counts())
print(data['carbody'].value_counts())
print(data['drivewheel'].value_counts())
print(data['enginelocation'].value_counts())
```

```
gas       185
diesel     20
Name: fueltype, dtype: int64
std      168
turbo     37
Name: aspiration, dtype: int64
four    115
two      90
Name: doornumber, dtype: int64
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: carbody, dtype: int64
fwd    120
rwd     76
4wd      9
Name: drivewheel, dtype: int64
front    202
rear       3
Name: enginelocation, dtype: int64
```

In [10]:
```
###data cleaning
```

In [11]:
```
#checking duplicates
data[data.duplicated()]
```

Out[11]:

symboling	make	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelb

0 rows × 25 columns

NO DUPLICATE VALUES

In [12]:
```
##checking null values
data.isnull().sum()
```

Out[12]:
```
symboling          0
make               0
fueltype           0
aspiration         0
doornumber         0
carbody            0
drivewheel         0
enginelocation     0
```

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

NO NULL VALUES

In [13]:
```
##we see that in make column car company and model given,splitting the company and m
data['car_company'] = data.make.str.split(' ').str.get(0).str.upper()
```

In [14]:
```
data.head()
```

Out[14]:

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

5 rows × 26 columns

In [15]:
```
data.car_company.unique()
```

Out[15]:
```
array(['ALFA-ROMERO', 'AUDI', 'BMW', 'CHEVROLET', 'DODGE', 'HONDA',
       'ISUZU', 'JAGUAR', 'MAXDA', 'MAZDA', 'BUICK', 'MERCURY',
       'MITSUBISHI', 'NISSAN', 'PEUGEOT', 'PLYMOUTH', 'PORSCHE',
       'PORCSHCE', 'RENAULT', 'SAAB', 'SUBARU', 'TOYOTA', 'TOYOUTA',
       'VOKSWAGEN', 'VOLKSWAGEN', 'VW', 'VOLVO'], dtype=object)
```

In [16]:
```
#VOLKSWAGEN has three different values as  VOKSWAGEN and VW
# MAZDA is also spelled as MAXDA
# PORSCHE as PORSCHE and PORCSCHE.
#TOYOTA AS TOYOUTA
```

In [17]:
```
data['car_company'] = data['car_company'].replace(['VOKSWAGEN','VW'],'VOLKSWAGEN')
data['car_company'] = data['car_company'].replace(['MAXDA'],'MAZDA')
data['car_company'] = data['car_company'].replace(['PORCSHCE'],'PORSCHE')
data['car_company'] = data['car_company'].replace(['TOYOUTA'],'TOYOTA')
```

In [18]:
```
data.car_company.unique()
```

Out[18]:
```
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 [19]:
```
data = data.drop(['make'], axis =1)
```

In [20]:
```
data.head()
```

Out[20]:

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase
0	3	gas	std	two	convertible	rwd	front	88.6
1	3	gas	std	two	convertible	rwd	front	88.6
2	1	gas	std	two	hatchback	rwd	front	94.5
3	2	gas	std	four	sedan	fwd	front	99.8
4	2	gas	std	four	sedan	4wd	front	99.4

5 rows × 25 columns

EXPLORATORY DATA ANALYSIS

In [21]:
```
import sweetviz
my_report = sweetviz.analyze([data,"Automobile"],target_feat='price')
```

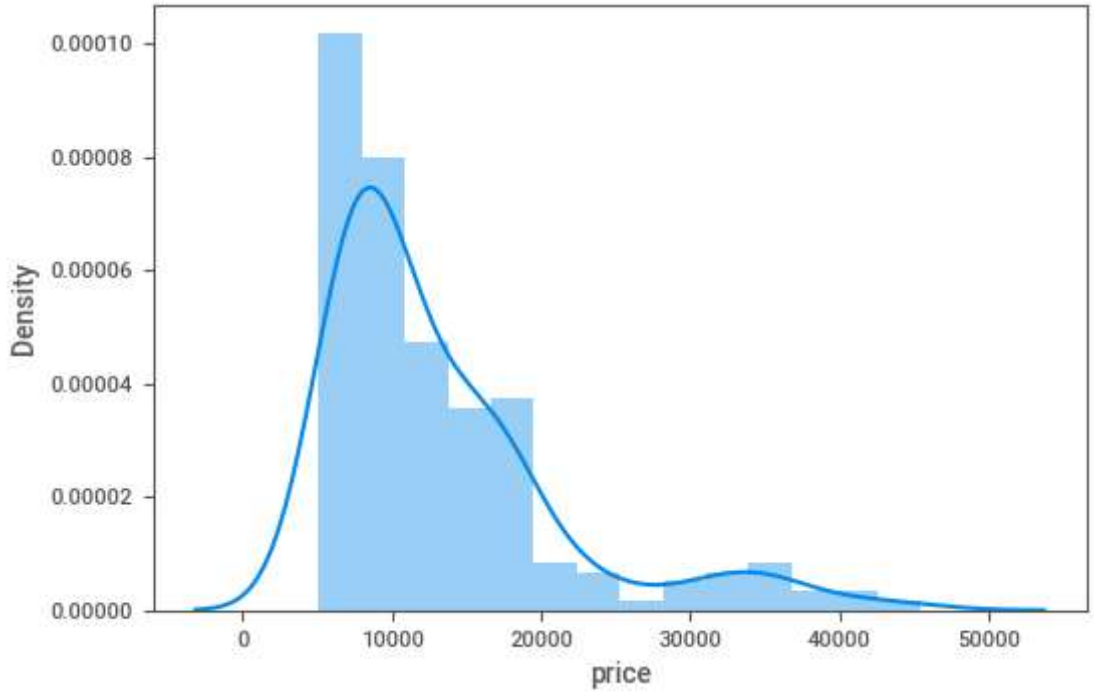
A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [22]:
```
my_report.show_html('eda.html')
```

```
Report eda.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up,
regardless, the report IS saved in your notebook/colab files.
```

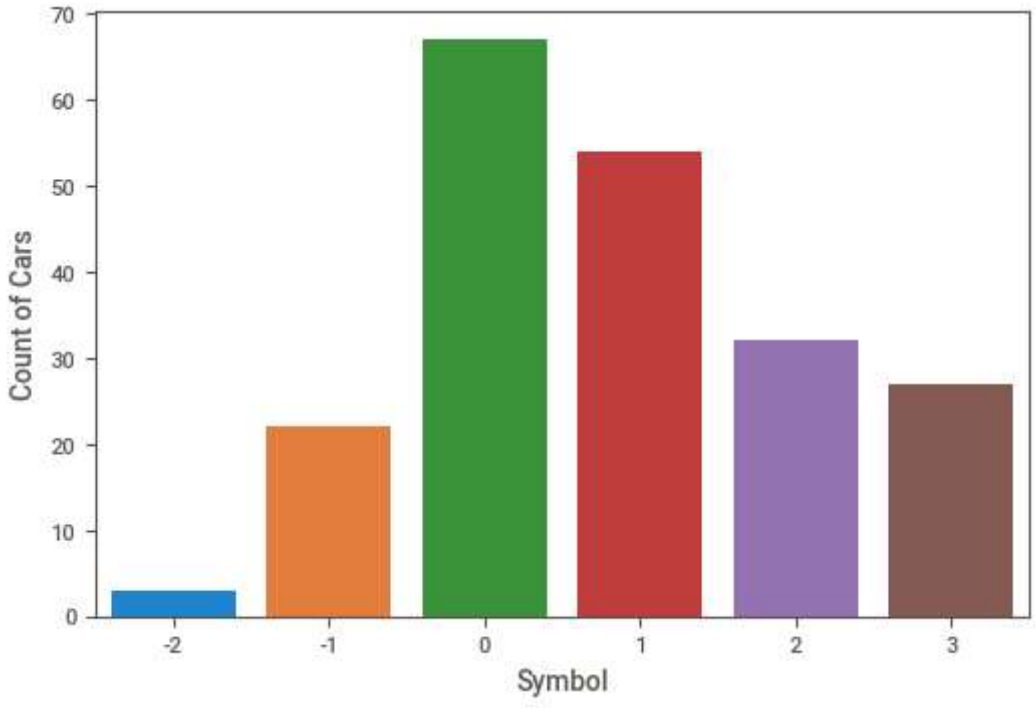
In [23]:
```
sns.distplot(data['price'])
```

Out[23]:
```
<AxesSubplot:xlabel='price', ylabel='Density'>
```



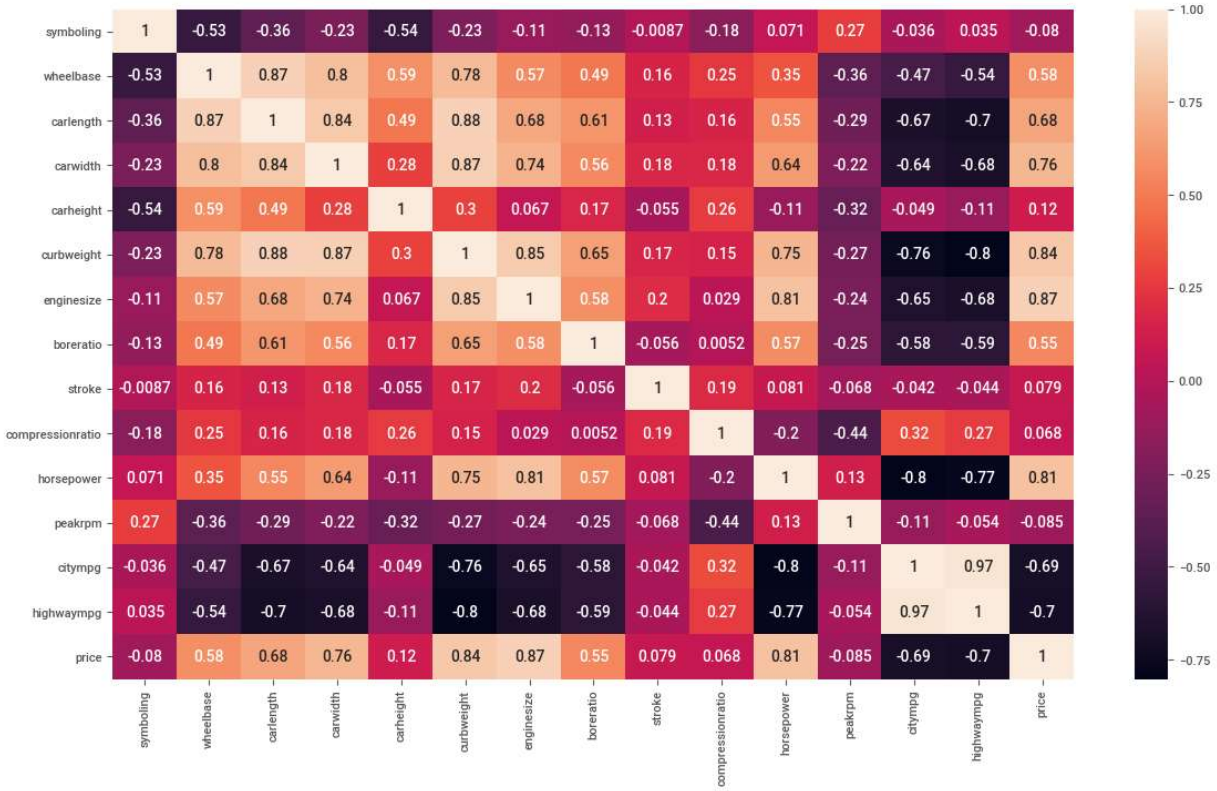
In [24]:
```
##symboling
ex = sns.countplot(data['symboling'])
ex.set(xlabel = 'Symbol',ylabel= 'Count of Cars')
```

Out[24]:
```
[Text(0.5, 0, 'Symbol'), Text(0, 0.5, 'Count of Cars')]
```



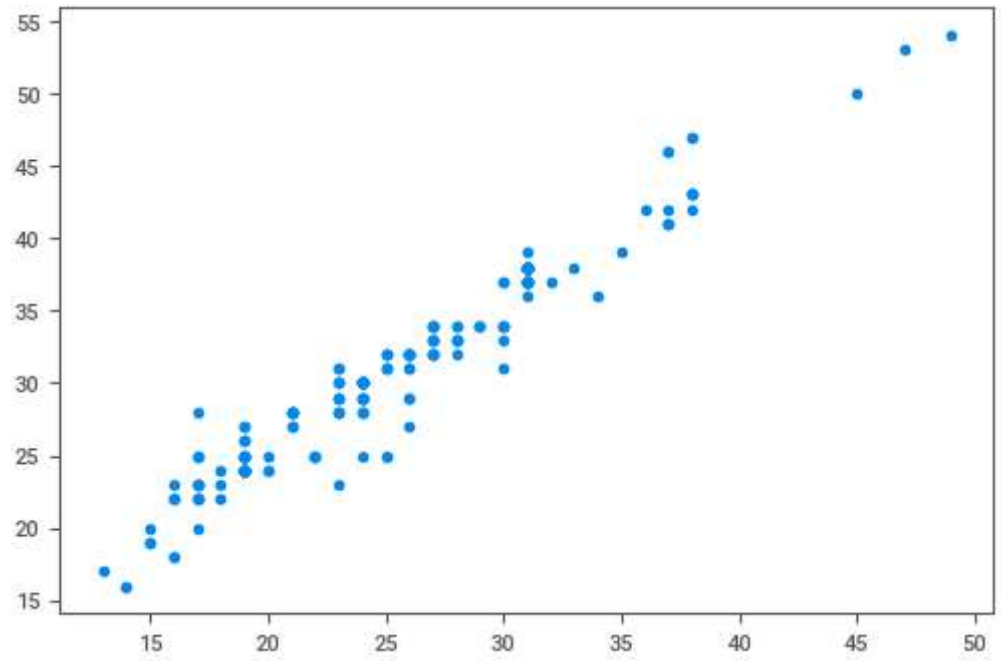
In [25]:
```
##correlation matrix
plt.figure(figsize=(14,8))
corr = data.corr()
sns.heatmap(corr,annot=True)
```

Out[25]:
```
<AxesSubplot:>
```

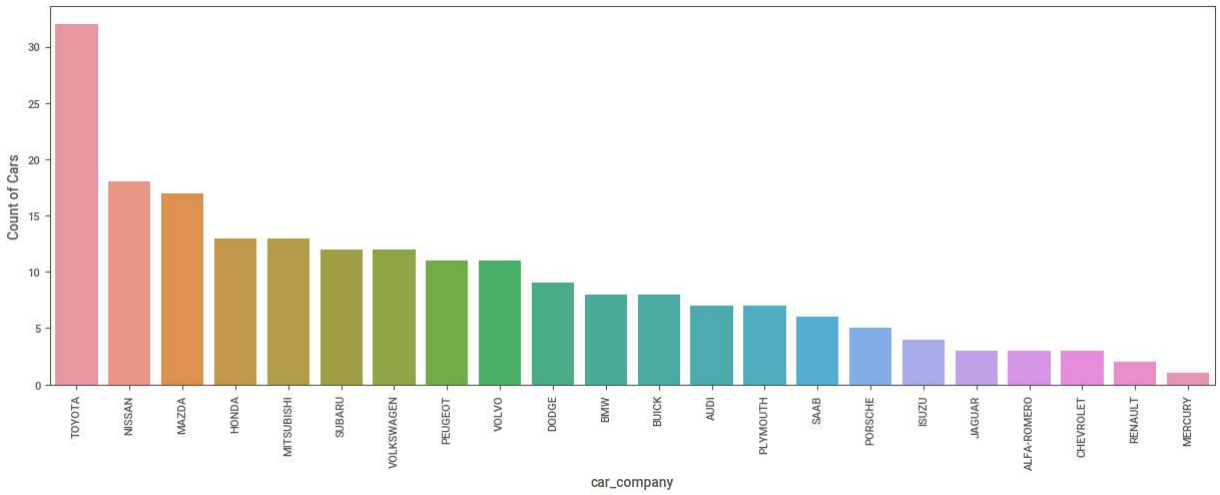


In [26]:
```
plt.scatter(data['citympg'],data['highwaympg'])
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x1f07b0d9dc0>
```

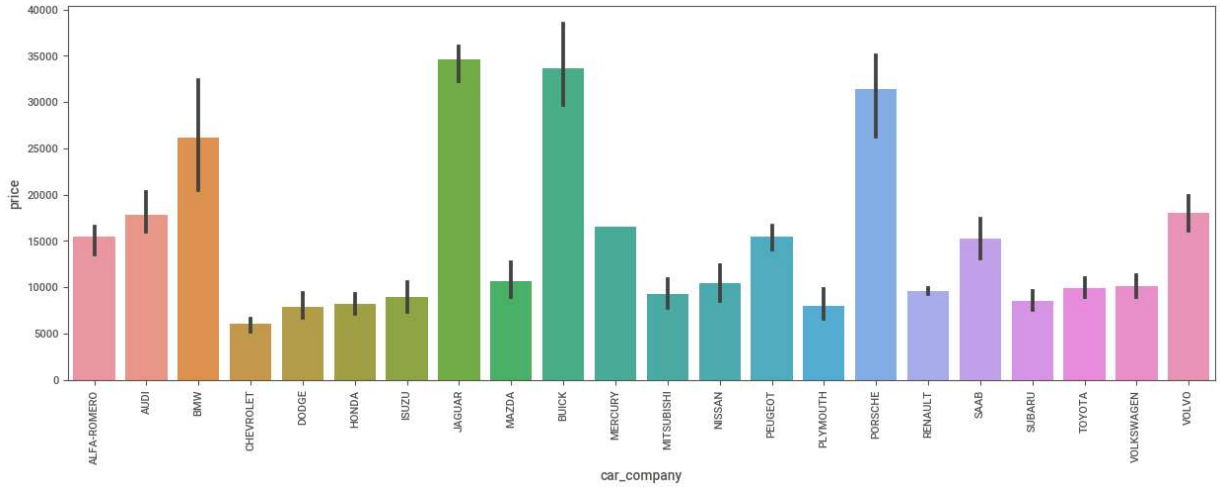


In [27]:
```
fig, ax = plt.subplots(figsize = (15,5))
plot = sns.countplot(data['car_company'], order=pd.value_counts(data['car_company']).
plot.set(xlabel = 'car_company', ylabel= 'Count of Cars')
plt.xticks(rotation=90)
plt.show()
```



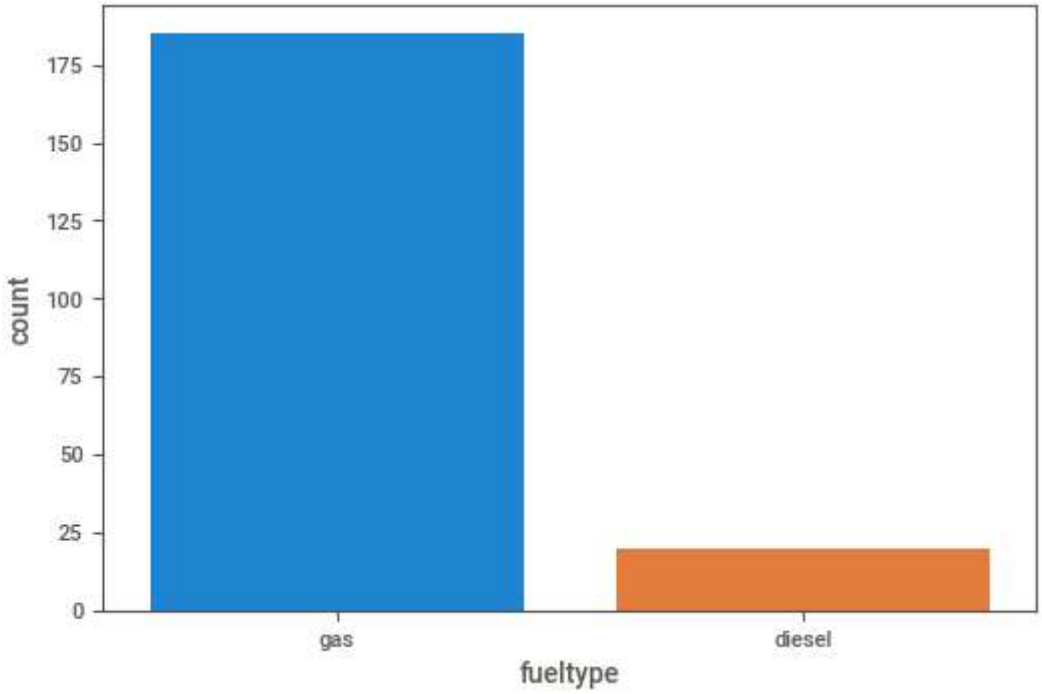
In [28]:
```
fig, ax = plt.subplots(figsize = (15,5))
sns.barplot('car_company','price',data=data)
plt.xticks(rotation=90)
```

Out[28]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21]),
 [Text(0, 0, 'ALFA-ROMERO'),
  Text(1, 0, 'AUDI'),
  Text(2, 0, 'BMW'),
  Text(3, 0, 'CHEVROLET'),
  Text(4, 0, 'DODGE'),
  Text(5, 0, 'HONDA'),
  Text(6, 0, 'ISUZU'),
  Text(7, 0, 'JAGUAR'),
  Text(8, 0, 'MAZDA'),
  Text(9, 0, 'BUICK'),
  Text(10, 0, 'MERCURY'),
  Text(11, 0, 'MITSUBISHI'),
  Text(12, 0, 'NISSAN'),
  Text(13, 0, 'PEUGEOT'),
  Text(14, 0, 'PLYMOUTH'),
  Text(15, 0, 'PORSCHE'),
  Text(16, 0, 'RENAULT'),
  Text(17, 0, 'SAAB'),
  Text(18, 0, 'SUBARU'),
  Text(19, 0, 'TOYOTA'),
  Text(20, 0, 'VOLKSWAGEN'),
  Text(21, 0, 'VOLVO')])
```

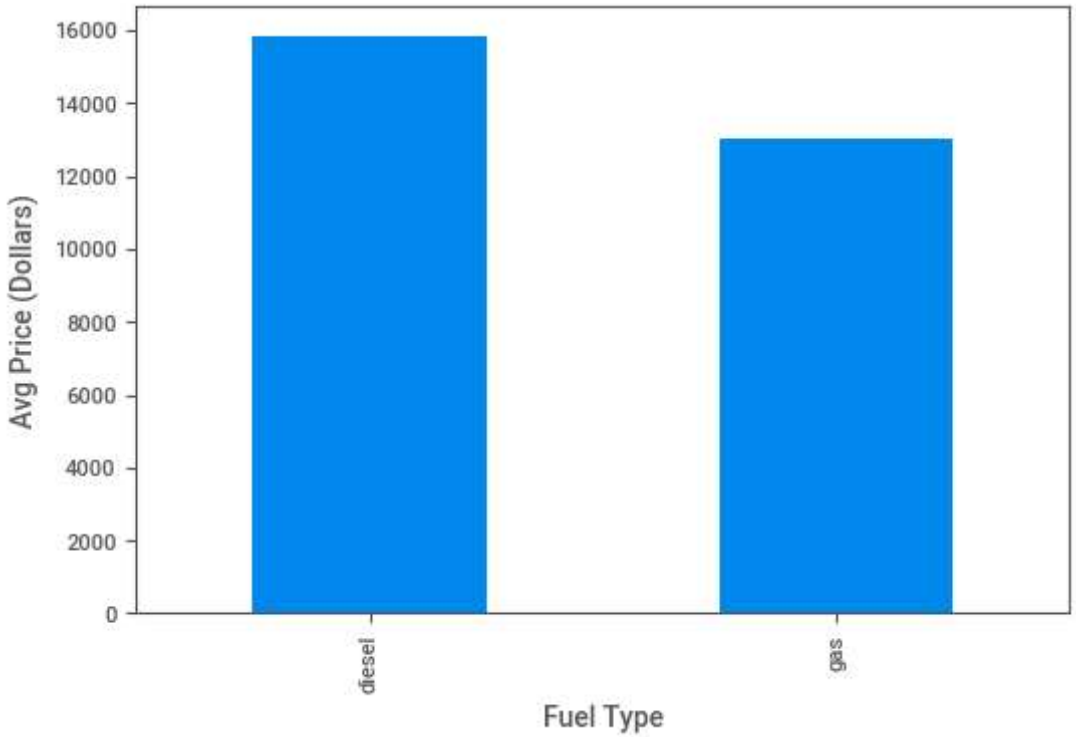


In [29]:
```
sns.countplot(data['fueltype'])
```

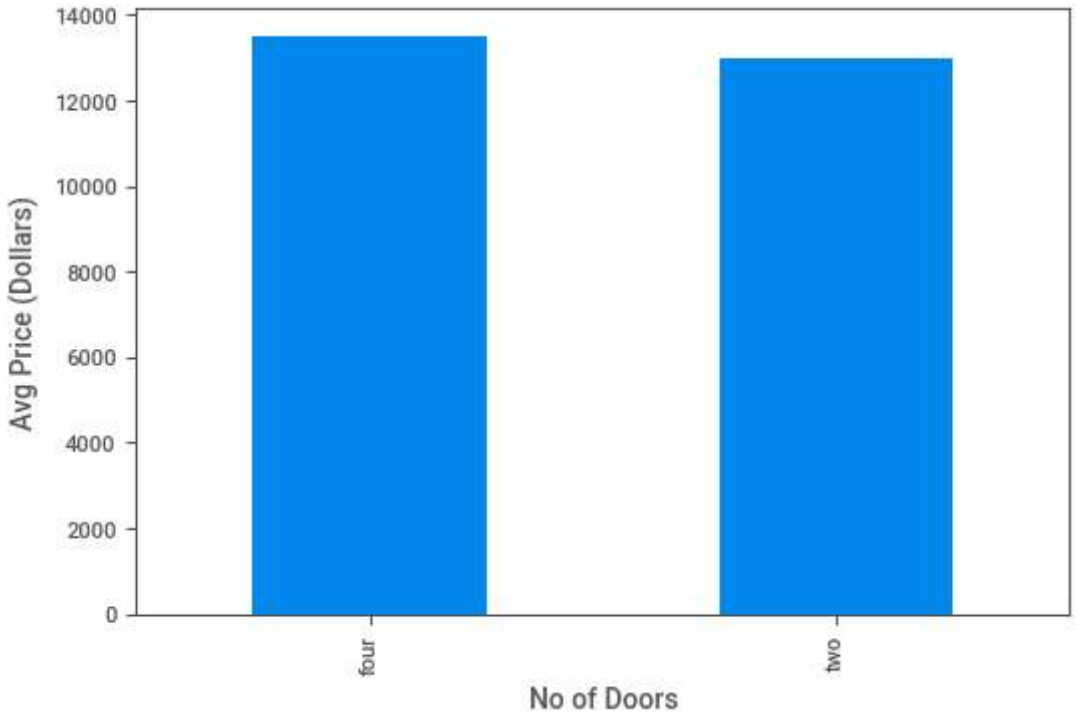
Out[29]:
```
<AxesSubplot:xlabel='fueltype', ylabel='count'>
```



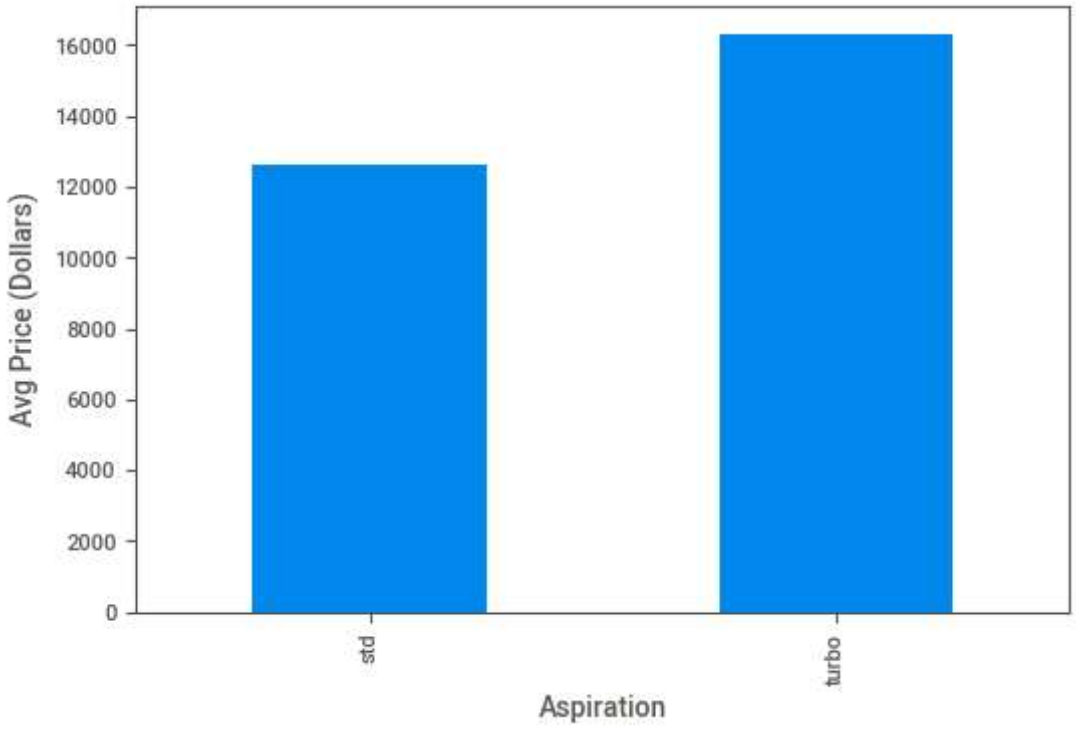
In [30]:
```
fuel_avg_price = data[['fueltype','price']].groupby("fueltype", as_index = False).me
plt1 = fuel_avg_price.plot(x = 'fueltype', kind='bar',legend = False, sort_columns =
plt1.set_xlabel("Fuel Type")
plt1.set_ylabel("Avg Price (Dollars)")
plt.show()
```



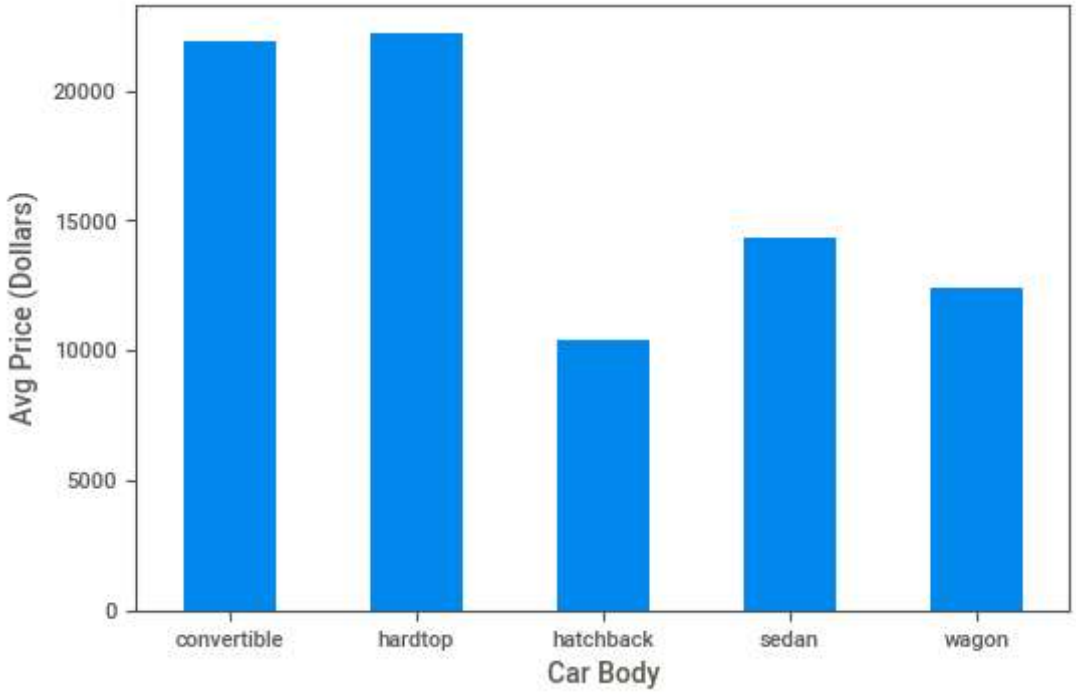
In [31]:
```
door_avg_price = data[['doornumber','price']].groupby("doornumber", as_index = False
plt1 = door_avg_price.plot(x = 'doornumber', kind='bar',legend = False, sort_columns
plt1.set_xlabel("No of Doors")
plt1.set_ylabel("Avg Price (Dollars)")
plt.show()
```



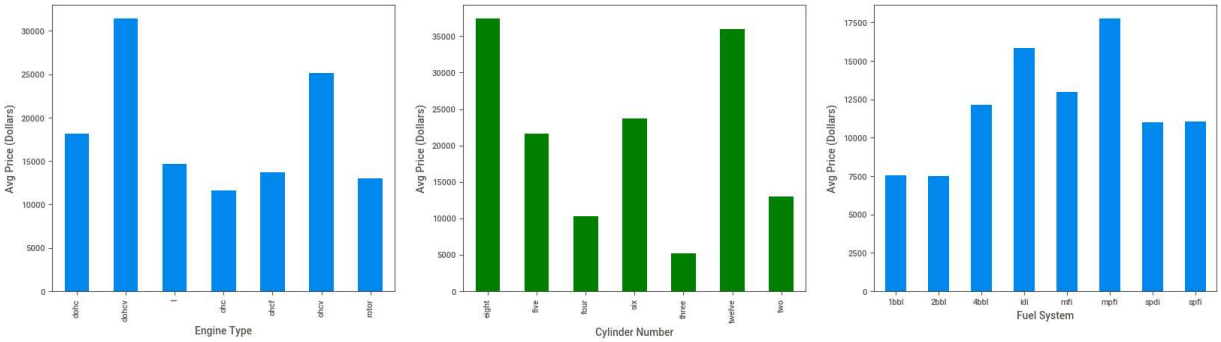
In [32]:
```
aspir_avg_price = data[['aspiration','price']].groupby("aspiration", as_index = Fals
plt1 = aspir_avg_price.plot(x = 'aspiration', kind='bar',legend = False, sort_column
plt1.set_xlabel("Aspiration")
plt1.set_ylabel("Avg Price (Dollars)")

plt.show()
```



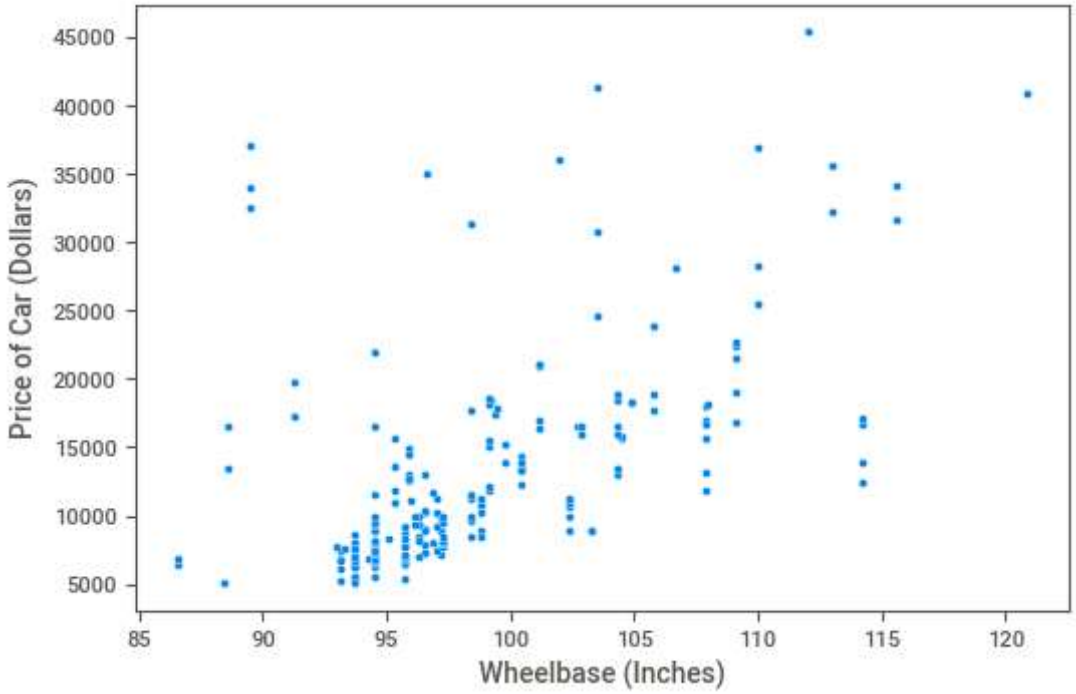
In [33]:
```
df_body_avg_price = data[['carbody','price']].groupby("carbody", as_index = False).m
plt1 = df_body_avg_price.plot(x = 'carbody', kind='bar',legend = False, sort_columns
plt1.set_xlabel("Car Body")
plt1.set_ylabel("Avg Price (Dollars)")
plt.xticks(rotation = 0)
plt.show()
```



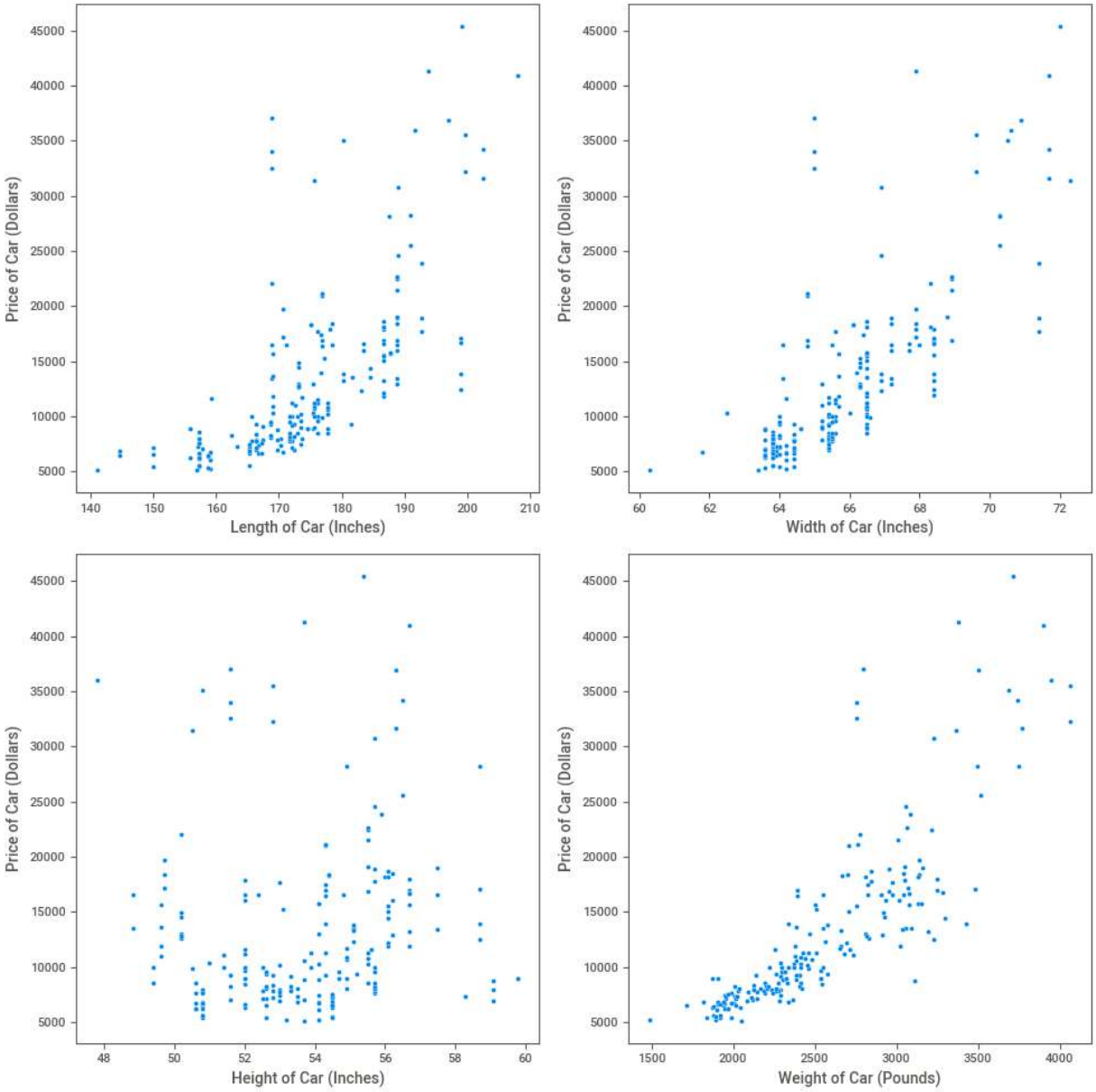
In [34]:
```
fig, axs = plt.subplots(1,3,figsize=(20,5))


df_engine_avg_price = data[['enginetype','price']].groupby("enginetype", as_index =
plt1 = df_engine_avg_price.plot(x = 'enginetype', kind='bar', sort_columns = True, l
plt1.set_xlabel("Engine Type")
plt1.set_ylabel("Avg Price (Dollars)")
plt.xticks(rotation = 0)
df_cylindernumber_avg_price = data[['cylindernumber','price']].groupby("cylindernumb
plt1 = df_cylindernumber_avg_price.plot(x = 'cylindernumber', kind='bar',color='g',
plt1.set_xlabel("Cylinder Number")
plt1.set_ylabel("Avg Price (Dollars)")
plt.xticks(rotation = 0)
df_fuelsystem_avg_price = data[['fuelsystem','price']].groupby("fuelsystem", as_inde
plt1 = df_fuelsystem_avg_price.plot(x = 'fuelsystem', kind='bar', sort_columns = Tru
plt1.set_xlabel("Fuel System")
plt1.set_ylabel("Avg Price (Dollars)")
plt.xticks(rotation = 0)
plt.show()
```



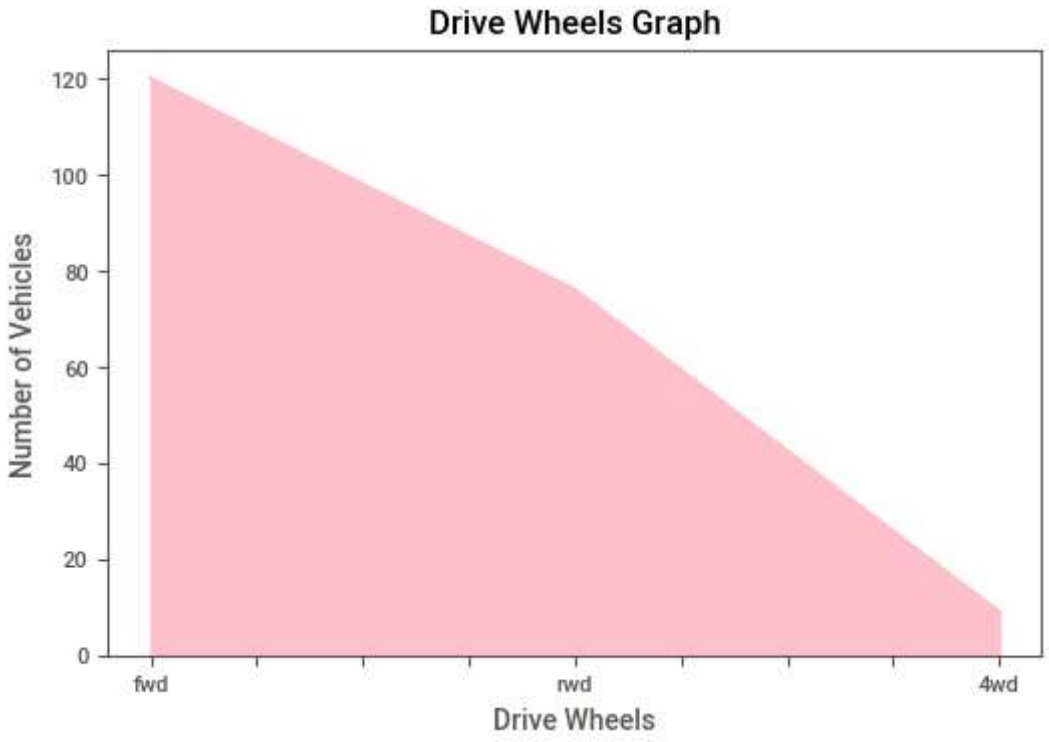
In [35]:
```
plt1 = sns.scatterplot(x = 'wheelbase', y = 'price', data = data)
plt1.set_xlabel('Wheelbase (Inches)')
plt1.set_ylabel('Price of Car (Dollars)')
plt.show()
```



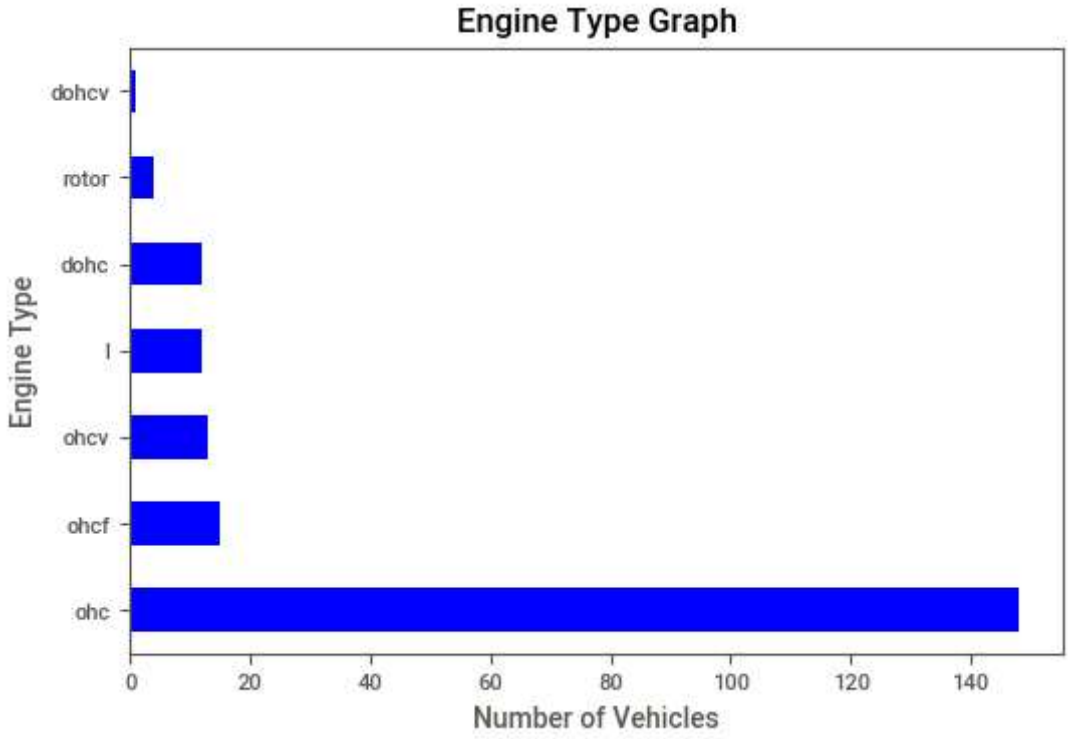
In [36]:
```
fig, axs = plt.subplots(2,2,figsize=(10,10))
plt1 = sns.scatterplot(x = 'carlength', y = 'price', data = data, ax = axs[0,0])
plt1.set_xlabel('Length of Car (Inches)')
plt1.set_ylabel('Price of Car (Dollars)')
plt2 = sns.scatterplot(x = 'carwidth', y = 'price', data = data, ax = axs[0,1])
plt2.set_xlabel('Width of Car (Inches)')
plt2.set_ylabel('Price of Car (Dollars)')
plt3 = sns.scatterplot(x = 'carheight', y = 'price', data = data, ax = axs[1,0])
plt3.set_xlabel('Height of Car (Inches)')
plt3.set_ylabel('Price of Car (Dollars)')
plt3 = sns.scatterplot(x = 'curbweight', y = 'price', data = data, ax = axs[1,1])
plt3.set_xlabel('Weight of Car (Pounds)')
plt3.set_ylabel('Price of Car (Dollars)')
plt.tight_layout()
```



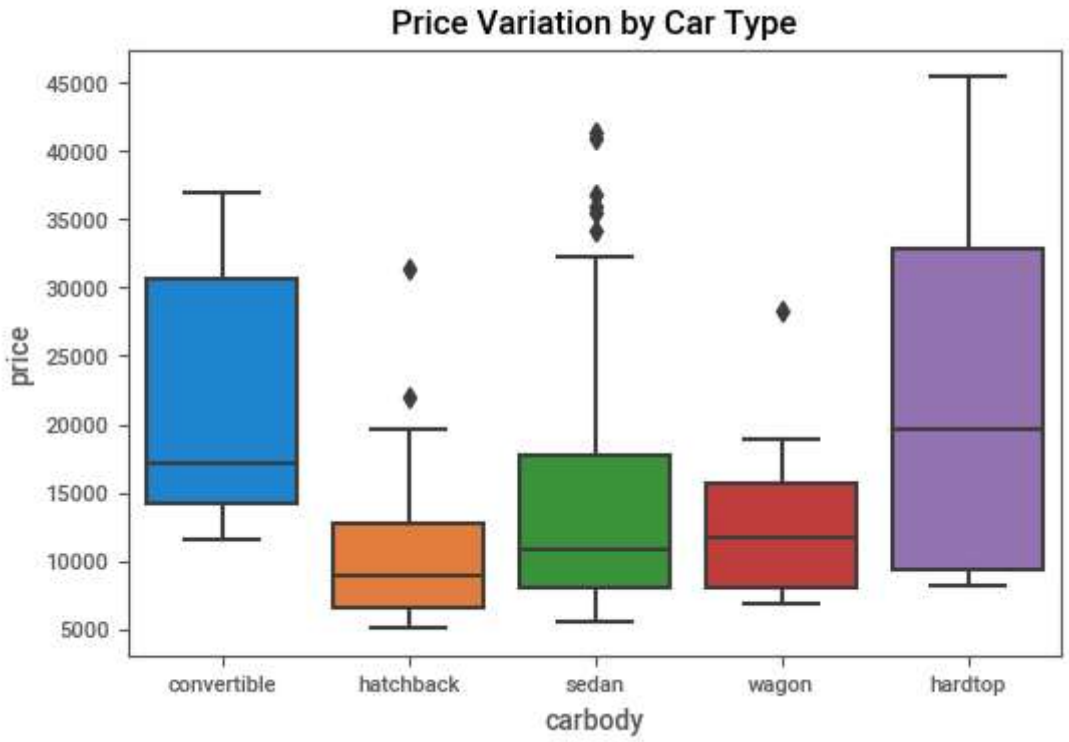
In [37]:
```
plt.title('Drive Wheels Graph')
plt.xlabel('Drive Wheels')
plt.ylabel('Number of Vehicles')
data['drivewheel'].value_counts().plot(kind='area',color='pink')
plt.show()
```



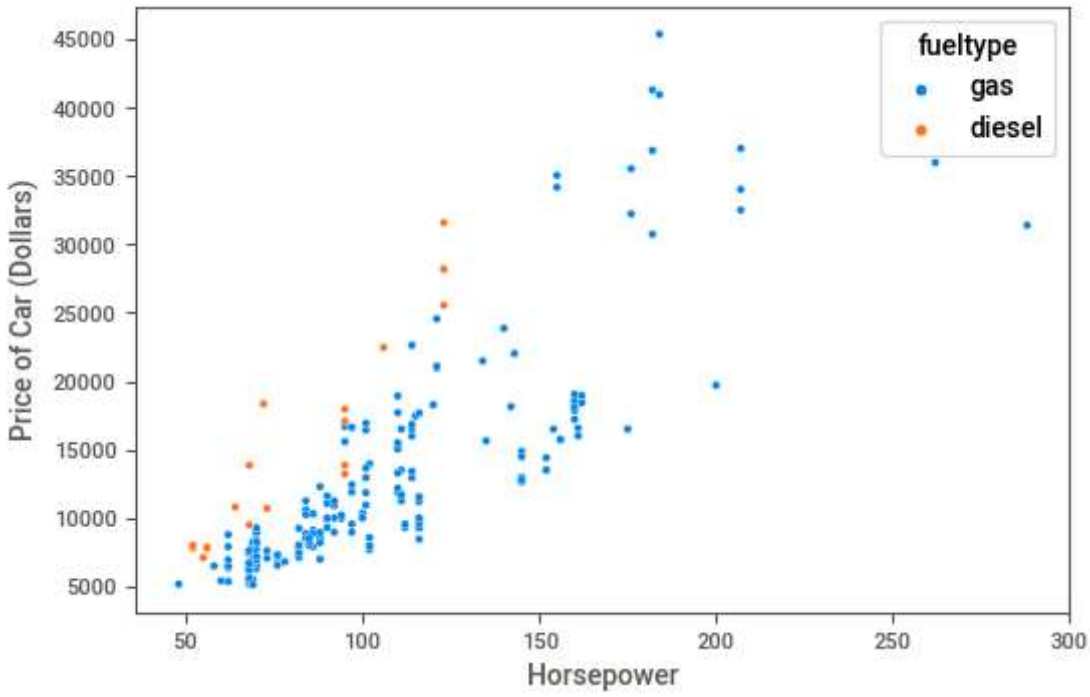
In [38]:
```
# Horizontal Bar Graph
plt.title('Engine Type Graph')
plt.ylabel('Engine Type')
plt.xlabel('Number of Vehicles')
data['enginetype'].value_counts().plot(kind='barh',color='b')
plt.show()
```



In [39]:
```
# Detecting outliers | Boxplot
plt.title('Price Variation by Car Type')
plt.xlabel('Car Type')
plt.ylabel('Price')
sns.boxplot(data['carbody'],data['price'])
plt.show()
```

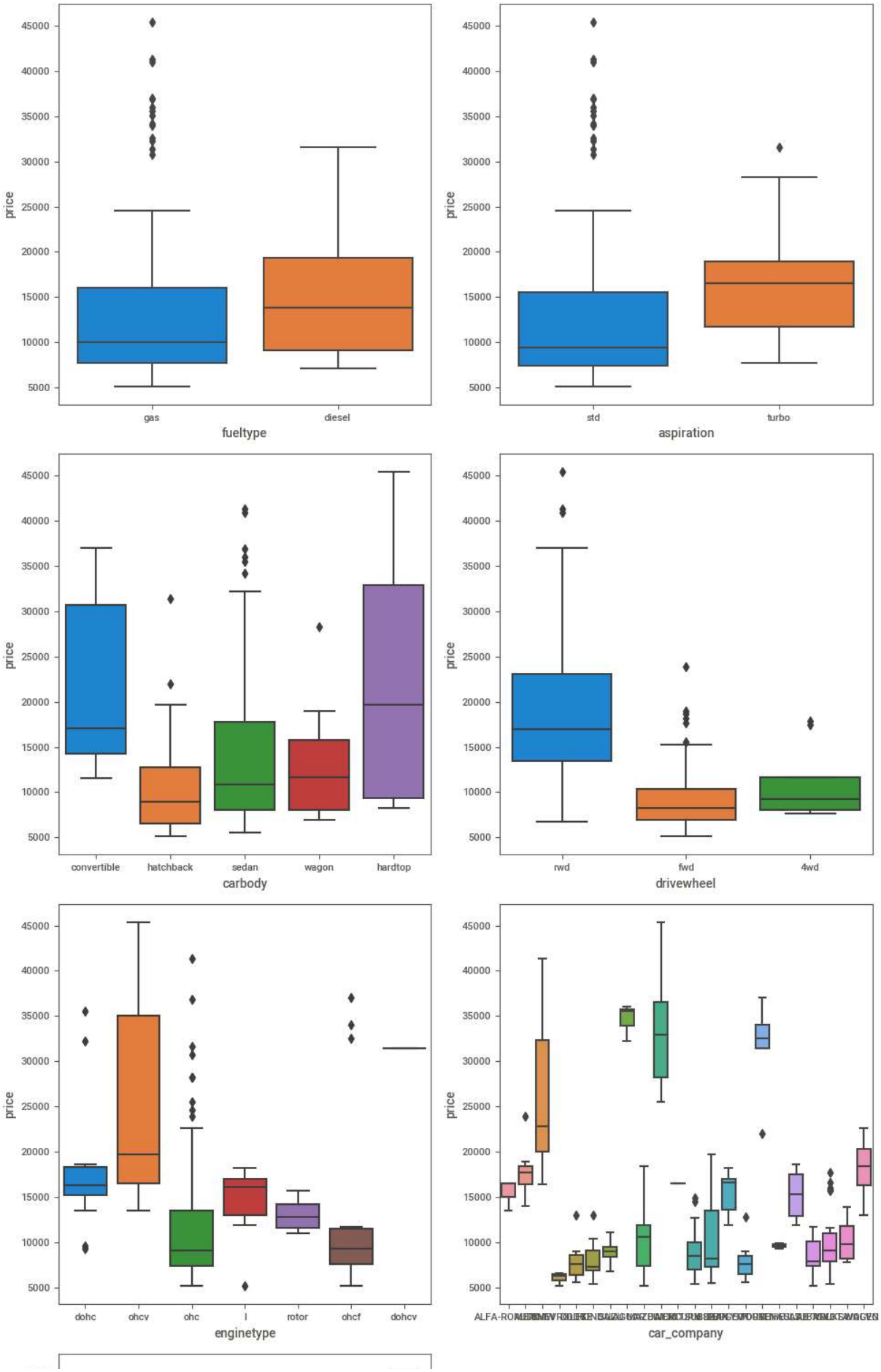


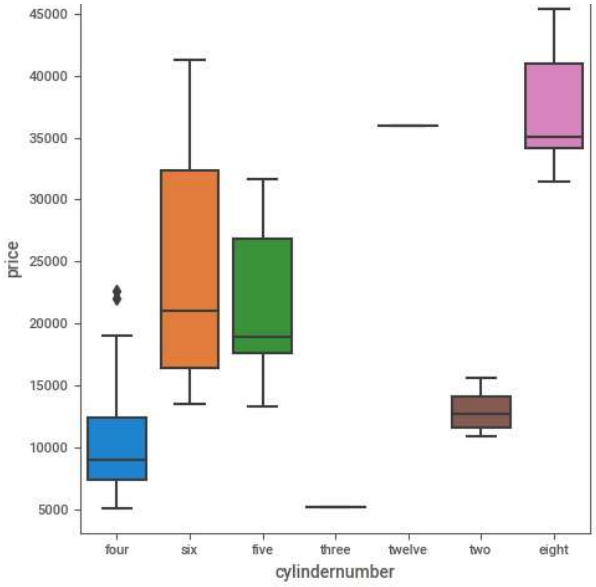
In [40]:
```
plt1 = sns.scatterplot(x = 'horsepower', y = 'price', hue = 'fueltype', data = data)
plt1.set_xlabel('Horsepower')
plt1.set_ylabel('Price of Car (Dollars)')
plt.show()
```



In [41]:
```
plt.figure(figsize=(10, 20))
plt.subplot(4,2,1)
sns.boxplot(x = 'fueltype', y = 'price', data = data)
plt.subplot(4,2,2)
sns.boxplot(x = 'aspiration', y = 'price', data = data)
plt.subplot(4,2,3)
sns.boxplot(x = 'carbody', y = 'price', data = data)
plt.subplot(4,2,4)
sns.boxplot(x = 'drivewheel', y = 'price', data = data)
plt.subplot(4,2,5)
sns.boxplot(x = 'enginetype', y = 'price', data = data)
```

```
plt.subplot(4,2,6)
sns.boxplot(x = 'car_company', y = 'price', data = data)
plt.subplot(4,2,7)
sns.boxplot(x = 'cylindernumber', y = 'price', data = data)
plt.tight_layout()
plt.show()
```





FEATURE ENGINEERING

In [42]:
```
# Identifying Categorical & Numerical Cols
cols = data.columns
num_cols = data._get_numeric_data().columns.to_list()
cat_cols = list(set(cols)-set(num_cols))

print('Numerical Columns')
print(num_cols)
print('\nCategorical Columns')
print(cat_cols)
```

```
Numerical Columns
['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight', 'engi
nesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citymp
g', 'highwaympg', 'price']

Categorical Columns
['fuelsystem', 'doornumber', 'enginetype', 'car_company', 'carbody', 'drivewheel',
'aspiration', 'enginelocation', 'cylindernumber', 'fueltype']
```

In [43]:
```
##Label encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for i in data[cat_cols]:
    data[i] = le.fit_transform(data[i])
```

In [44]:
```
data.head()
```

Out[44]:

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	‹
0	3	1	0	1	0	2	0	88.6	
1	3	1	0	1	0	2	0	88.6	
2	1	1	0	1	2	2	0	94.5	
3	2	1	0	0	3	1	0	99.8	
4	2	1	0	0	3	0	0	99.4	

5 rows × 25 columns

In [45]:
```
data.describe()
```

Out[45]:

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wl
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	20
mean	0.834146	0.902439	0.180488	0.439024	2.614634	1.326829	0.014634	9
std	1.245307	0.297446	0.385535	0.497483	0.859081	0.556171	0.120377	
min	-2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	8
25%	0.000000	1.000000	0.000000	0.000000	2.000000	1.000000	0.000000	9
50%	1.000000	1.000000	0.000000	0.000000	3.000000	1.000000	0.000000	9
75%	2.000000	1.000000	0.000000	1.000000	3.000000	2.000000	0.000000	10
max	3.000000	1.000000	1.000000	1.000000	4.000000	2.000000	1.000000	12

8 rows × 25 columns

In [46]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
```

In [47]:
```
# train-test split
X = data['horsepower']
y = data['price']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.3,random_state=42)

print(X.head())
print(y.head())
```

```
0    111
1    111
2    154
3    102
4    115
Name: horsepower, dtype: int64
0    13495.0
1    16500.0
2    16500.0
3    13950.0
4    17450.0
Name: price, dtype: float64
```

In [48]:
```
# As data is in 1D array, need to convert to 2D array for linear regression
X_train = X_train.values.reshape(-1,1)
X_test = X_test.values.reshape(-1,1)
```

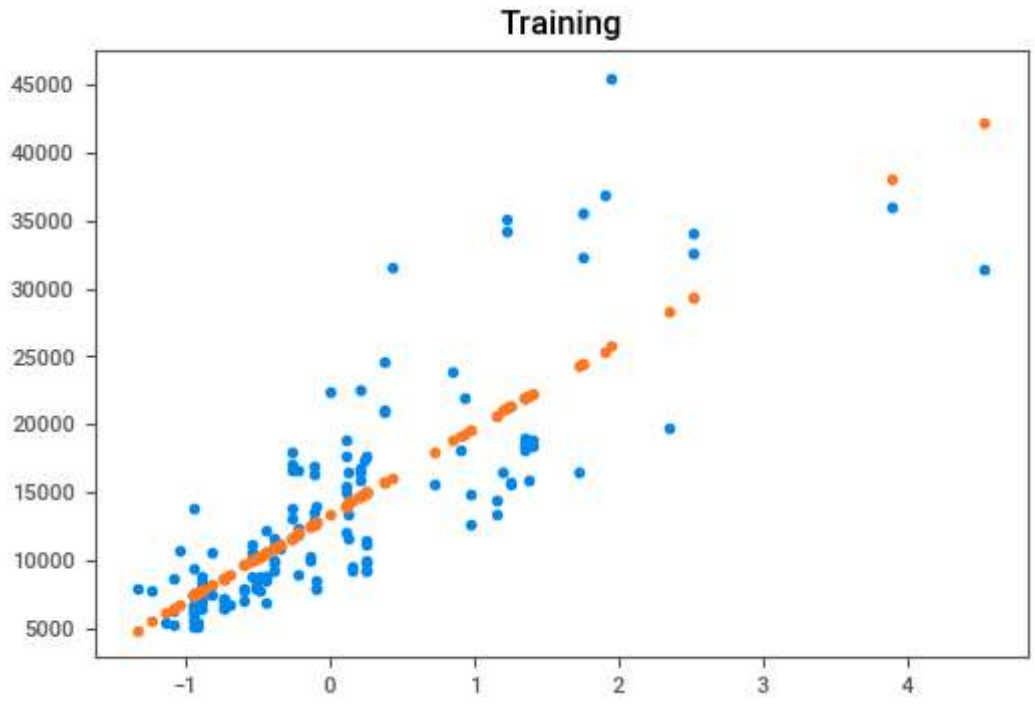
In [49]:
```
# Scaling
# Since Price & Horse Power have large gaps in terms of value, Applying Standard Sca
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

X_train = ss.fit_transform(X_train)
X_test = ss.fit_transform(X_test)
```

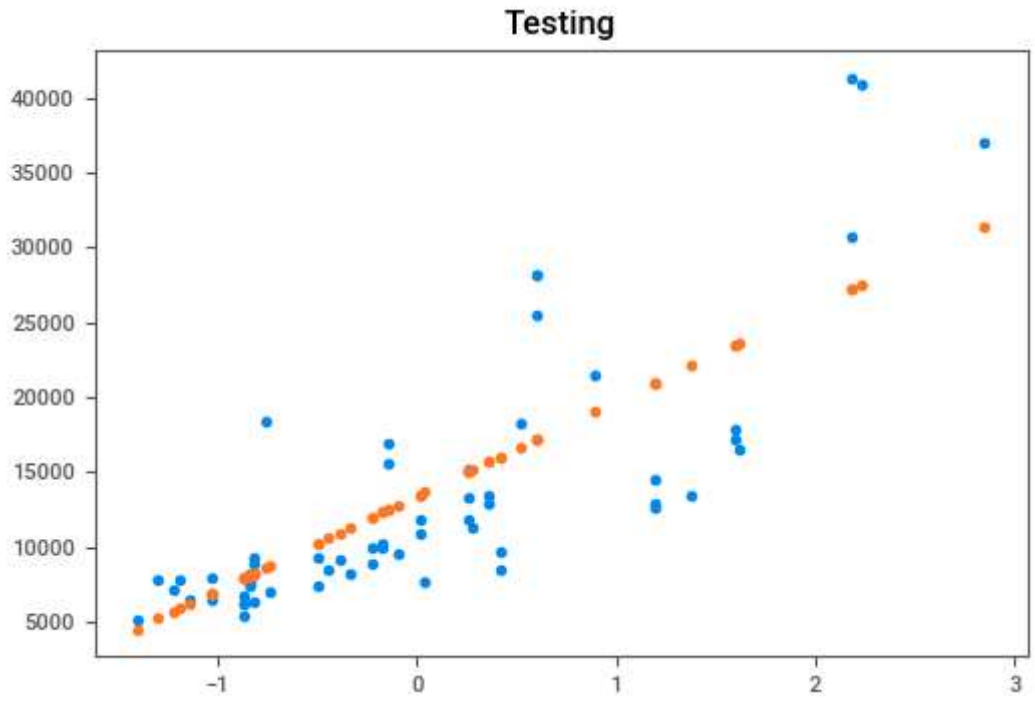
In [50]:
```
# Linear regression fit
lr = LinearRegression()
lr.fit(X_train,y_train)
```

```
print('Intercept is',lr.intercept_)
print('Coefficient is',lr.coef_)
```

```
Intercept is 13408.503496503494
Coefficient is [6332.93047557]
```

In [51]:
```
# Predictions
# Blue scatter -> actual-training data
# Orange scatter -> predicted data

# Training
y_train_pred = lr.predict(X_train)
plt.title('Training')
plt.scatter(X_train,y_train)
plt.scatter(X_train,y_train_pred)
plt.show()

# Testing
y_test_pred = lr.predict(X_test)
plt.title('Testing')
plt.scatter(X_test,y_test)
plt.scatter(X_test,y_test_pred)
plt.show()
```





In [52]:
```
# Let's check how much good fit it is by calculating R-Squared

print('Mean Squared Error for training data is',mean_squared_error(y_train,y_train_p
print('R2 Score for training data in LR is',r2_score(y_train,y_train_pred))
print('\n')

print('Mean Squared Error for testing data is',mean_squared_error(y_test,y_test_pred
print('R2 Score for testing data in LR is',r2_score(y_test,y_test_pred))
```

```
Mean Squared Error for training data is 20843608.761176858
R2 Score for training data in LR is 0.6580190372125265


Mean Squared Error for testing data is 24492365.07282414
R2 Score for testing data in LR is 0.6464951326151097
```

In [53]:
```
##scaling the numeric features
from sklearn.preprocessing import StandardScaler
```

In [54]:
```
from sklearn.feature_selection import VarianceThreshold
```

In [55]:
```
X=data.drop(labels=['price'],axis=1)
```

In [56]:
```
X.head()
```

Out[56]:

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	c
0	3	1	0	1	0	2	0	88.6	
1	3	1	0	1	0	2	0	88.6	
2	1	1	0	1	2	2	0	94.5	
3	2	1	0	0	3	1	0	99.8	
4	2	1	0	0	3	0	0	99.4	

5 rows × 24 columns

In [57]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=101)
X_train.shape,X_test.shape
```

Out[57]:
```
((143, 24), (62, 24))
```

In [58]:
```
from sklearn.feature_selection import mutual_info_regression
```

In [59]:
```
mutual_info=mutual_info_regression(X_train,y_train)
mutual_info
```

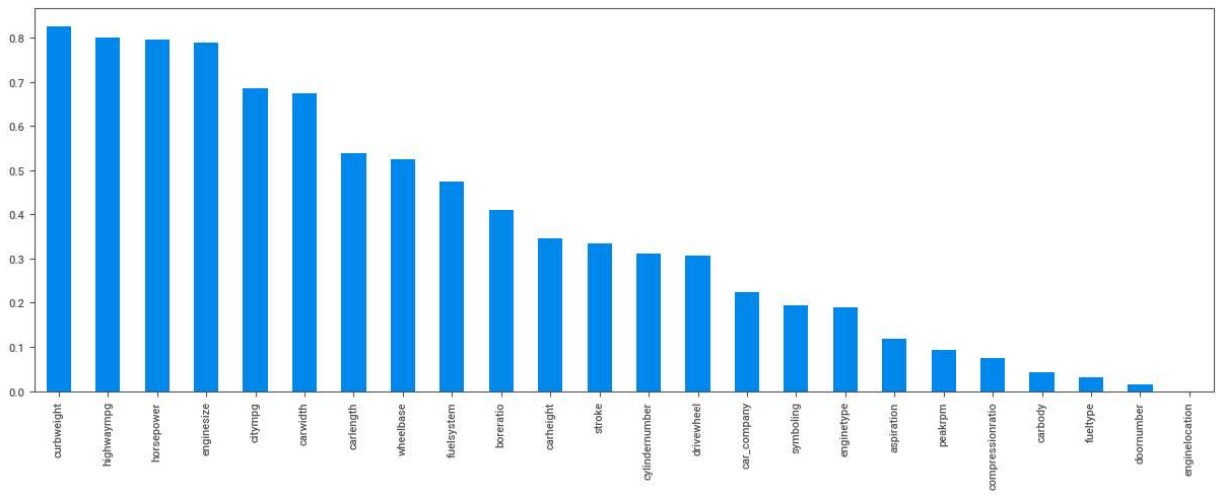
Out[59]:
```
array([0.19358099, 0.03143474, 0.11901995, 0.01567695, 0.04196255,
       0.3061141 , 0.        , 0.52473734, 0.53764079, 0.67486569,
       0.34576153, 0.82557761, 0.19052075, 0.31178017, 0.7893916 ,
       0.47409013, 0.40977795, 0.33488603, 0.07388889, 0.79512127,
       0.09308696, 0.68605719, 0.79932331, 0.22387928])
```

In [60]:
```
mutual_info=pd.Series(mutual_info)
mutual_info.index=X_train.columns
mutual_info.sort_values(ascending=False)
```

Out[60]:
```
curbweight          0.825578
highwaympg          0.799323
horsepower          0.795121
enginesize          0.789392
citympg             0.686057
carwidth            0.674866
carlength           0.537641
wheelbase           0.524737
fuelsystem          0.474090
boreratio           0.409778
carheight           0.345762
stroke              0.334886
cylindernumber      0.311780
drivewheel          0.306114
car_company         0.223879
symboling           0.193581
enginetype          0.190521
aspiration          0.119020
peakrpm             0.093087
compressionratio    0.073889
carbody             0.041963
fueltype            0.031435
doornumber          0.015677
enginelocation      0.000000
dtype: float64
```

In [61]:
```
mutual_info.sort_values(ascending=False).plot.bar(figsize=(15,5))
```

Out[61]:
```
<AxesSubplot:>
```



In [62]:
```
from sklearn.feature_selection import SelectKBest
```

In [63]:
```
##now we select the top 10 imp features
sel_ten_cols = SelectKBest(mutual_info_regression,k=10)
sel_ten_cols.fit(X_train,y_train)
X_train.columns[sel_ten_cols.get_support()]
```

Out[63]:
```
Index(['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize',
       'fuelsystem', 'boreratio', 'horsepower', 'citympg', 'highwaympg'],
      dtype='object')
```

In [64]:
```
X_features=X[['wheelbase', 'carlength', 'carwidth', 'curbweight', 'enginesize',
       'fuelsystem', 'boreratio', 'horsepower', 'citympg', 'highwaympg']]
```

In [65]:
```
X_features.head()
```

Out[65]:

	wheelbase	carlength	carwidth	curbweight	enginesize	fuelsystem	boreratio	horsepower	city
0	88.6	168.8	64.1	2548	130	5	3.47	111	
1	88.6	168.8	64.1	2548	130	5	3.47	111	
2	94.5	171.2	65.5	2823	152	5	2.68	154	
3	99.8	176.6	66.2	2337	109	5	3.19	102	
4	99.4	176.6	66.4	2824	136	5	3.19	115	

In [66]:
```
X_train_f,X_test_f,y_train,y_test=train_test_split(X_features,y,test_size=0.3,random
X_train_f.shape,X_test_f.shape
```

Out[66]:
```
((143, 10), (62, 10))
```

In [67]:
```
lr.fit(X_train_f,y_train)
y_pred_f=lr.predict(X_test_f)
```

In [68]:
```
print('Mean Squared Error for testing data is',mean_squared_error(y_test,y_pred_f))
print('R2 Score for testing data in LR is',r2_score(y_test,y_pred_f))
```

```
Mean Squared Error for testing data is 16258191.51616723
R2 Score for testing data in LR is 0.7653411657570832
```

In [69]:
```
scaler=StandardScaler()
```

In [70]:
```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [71]:
```
from sklearn.decomposition import PCA

pca = PCA()
```

In [72]:
```
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

In [73]:
```
pca.explained_variance_ratio_
```

Out[73]:
```
array([3.18441012e-01, 1.86313722e-01, 9.47407760e-02, 6.41294318e-02,
       5.41606783e-02, 4.67038821e-02, 3.86339669e-02, 3.51085999e-02,
       2.74097884e-02, 2.29736425e-02, 2.09937344e-02, 1.72155330e-02,
       1.48719664e-02, 1.37510741e-02, 1.13929660e-02, 9.11907665e-03,
```

```
       7.13410453e-03, 5.97989705e-03, 4.28344212e-03, 2.51628267e-03,
       2.21025445e-03, 1.23221993e-03, 5.06724349e-04, 1.77224814e-04])
```

In [74]:
```
pca = PCA(n_components=5)

X_train_5 = pca.fit_transform(X_train_scaled)
X_test_5 = pca.transform(X_test_scaled)
```

In [75]:
```
lr.fit(X_train_5,y_train)
```

Out[75]:
```
LinearRegression()
```

In [76]:
```
y_pred = lr.predict(X_test_5)
```

In [77]:
```
print('Mean Squared Error for testing data is',mean_squared_error(y_test,y_pred))
print('R2 Score for testing data in LR is',r2_score(y_test,y_pred))
```

```
Mean Squared Error for testing data is 66749275.64663581
R2 Score for testing data in LR is 0.03658982032392699
```

In [78]:
```
from sklearn.linear_model import SGDRegressor
```

In [79]:
```
sgd_regressor=SGDRegressor(alpha=0.0005,learning_rate='optimal',eta0=0.001)
```

In [80]:
```
sgd_regressor.fit(X_train_f,y_train)
```

Out[80]:
```
SGDRegressor(alpha=0.0005, eta0=0.001, learning_rate='optimal')
```

In [81]:
```
sgd_pred = sgd_regressor.predict(X_test_f)
```

In [82]:
```
mean_squared_error(sgd_pred,y_test)
```

Out[82]:
```
1.3841255424262724e+35
```

In [83]:
```
r2_score(sgd_pred,y_test)
```

Out[83]:
```
-25.0781989148036
```

In [84]:
```
X_feature_scaled=scaler.fit_transform(X_features)
```

In [85]:
```
X_feature_scaled=pd.DataFrame(X_feature_scaled)
```

In [86]:
```
X_feature_scaled.head()
```

Out[86]:

	0	1	2	3	4	5	6	7	8	
0	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.869568	0.519071	0.174483	-0.646553	-
1	-1.690772	-0.426521	-0.844782	-0.014566	0.074449	0.869568	0.519071	0.174483	-0.646553	-
2	-0.708596	-0.231513	-0.190566	0.514882	0.604046	0.869568	-2.404880	1.264536	-0.953012	-
3	0.173698	0.207256	0.136542	-0.420797	-0.431076	0.869568	-0.517266	-0.053668	-0.186865	-
4	0.107110	0.207256	0.230001	0.516807	0.218885	0.869568	-0.517266	0.275883	-1.106241	-

In [87]:
```
X_train_s,X_test_s,y_train,y_test=train_test_split(X_feature_scaled,y,test_size=0.3,
X_train_s.shape,X_test_s.shape
```

Out[87]: ((143, 10), (62, 10))

In [88]:
```
SGDRegressor().fit(X_train_s,y_train)
```

Out[88]: SGDRegressor()

In [89]:
```
pred=sgd_regressor.predict(X_test_s)
```

In [90]:
```
print('Mean Squared Error for testing data is',mean_squared_error(y_test,pred))
print('R2 Score for testing data in sgdregressor is',r2_score(y_test,pred))
```

```
Mean Squared Error for testing data is 1.2938842253323437e+28
R2 Score for testing data in sgdregressor is -2.1432349505816796e+20
```

In [91]:
```
lr.fit(X_train_s,y_train)
```

Out[91]: LinearRegression()

In [92]:
```
predicts=lr.predict(X_test_s)
```

In [93]:
```
print('Mean Squared Error for testing data is',mean_squared_error(y_test,predicts))
print('R2 Score for testing data in LR is',r2_score(y_test,predicts))
```

```
Mean Squared Error for testing data is 11511898.099945687
R2 Score for testing data in LR is 0.8093129054958448
```

In [94]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn import svm
```

In [95]:
```
models = []
models.append(('DTR', DecisionTreeRegressor()))
models.append(('svr', svm.SVR()))
models.append(('RFr', RandomForestRegressor()))
```

In [96]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

In [97]:
```
results = []
for name,model in models:
    kfold = KFold(n_splits=10,random_state=1,shuffle=True)
    cvresults = cross_val_score(model,X_train_s,y_train,cv=kfold)
    results.append(cvresults)
    output = "%s: %f(%f)" % (name,cvresults.mean(),cvresults.std())
    print(output)
```

```
DTR: 0.795101(0.097482)
svr: -0.204919(0.223145)
RFr: 0.883254(0.063268)
```

In [98]:
```
rfr=RandomForestRegressor(n_estimators=50,max_depth=5,min_samples_leaf=5,min_samples
```

In [99]:
```
rfr.fit(X_train_s,y_train)
```

Out[99]:
```
RandomForestRegressor(max_depth=5, min_samples_leaf=5, min_samples_split=4,
                      n_estimators=50, n_jobs=1, random_state=1)
```

In [100…
```
rfr_pred=rfr.predict(X_test_s)
```

In [101…
```
print('Mean Squared Error for testing data is',mean_squared_error(y_test,rfr_pred))
print('R2 Score for testing data in random forest regressor is',r2_score(y_test,rfr_
```

```
Mean Squared Error for testing data is 6103456.741882865
R2 Score for testing data in random forest regressor is 0.8989002141578258
```

In [102…
```
X_features.head()
```

Out[102…

	wheelbase	carlength	carwidth	curbweight	enginesize	fuelsystem	boreratio	horsepower	city
0	88.6	168.8	64.1	2548	130	5	3.47	111	
1	88.6	168.8	64.1	2548	130	5	3.47	111	
2	94.5	171.2	65.5	2823	152	5	2.68	154	
3	99.8	176.6	66.2	2337	109	5	3.19	102	
4	99.4	176.6	66.4	2824	136	5	3.19	115	

In [103…
```
X_features=X_features.drop('highwaympg',axis=1)
```

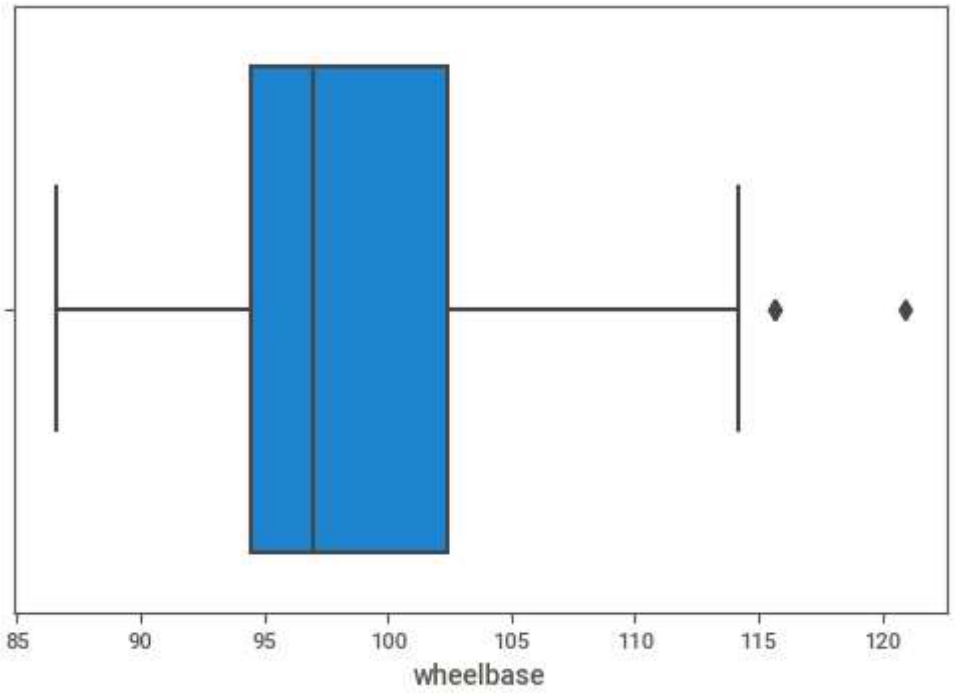
In [104…
```
X_features.head()
```

Out[104…

	wheelbase	carlength	carwidth	curbweight	enginesize	fuelsystem	boreratio	horsepower	city
0	88.6	168.8	64.1	2548	130	5	3.47	111	
1	88.6	168.8	64.1	2548	130	5	3.47	111	
2	94.5	171.2	65.5	2823	152	5	2.68	154	
3	99.8	176.6	66.2	2337	109	5	3.19	102	
4	99.4	176.6	66.4	2824	136	5	3.19	115	

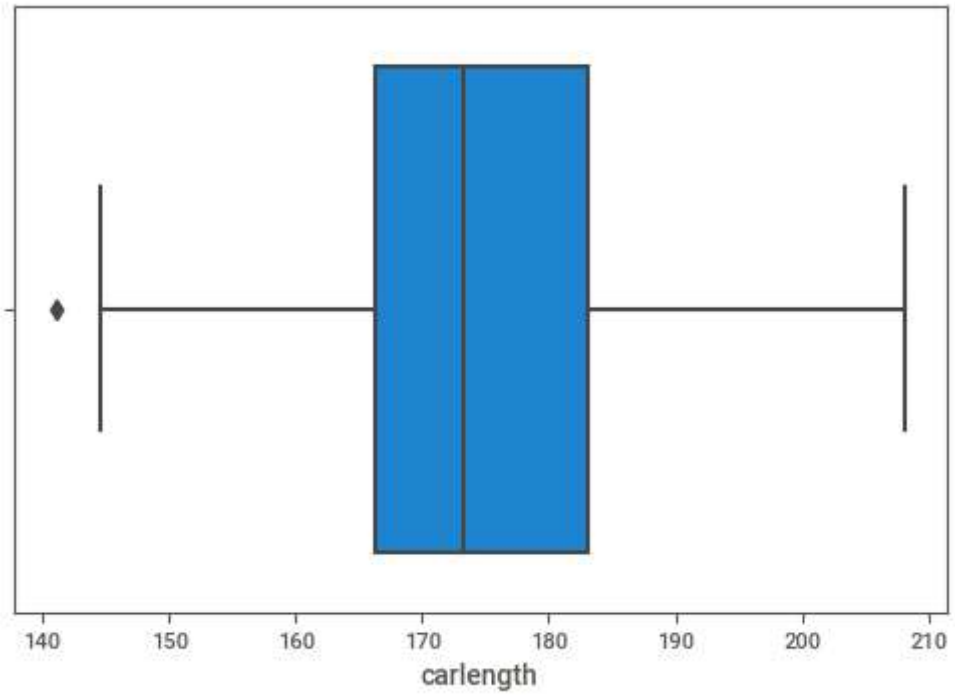
In [105…
```
sns.boxplot('wheelbase',data=X_features,orient='h')
```

Out[105… `<AxesSubplot:xlabel='wheelbase'>`



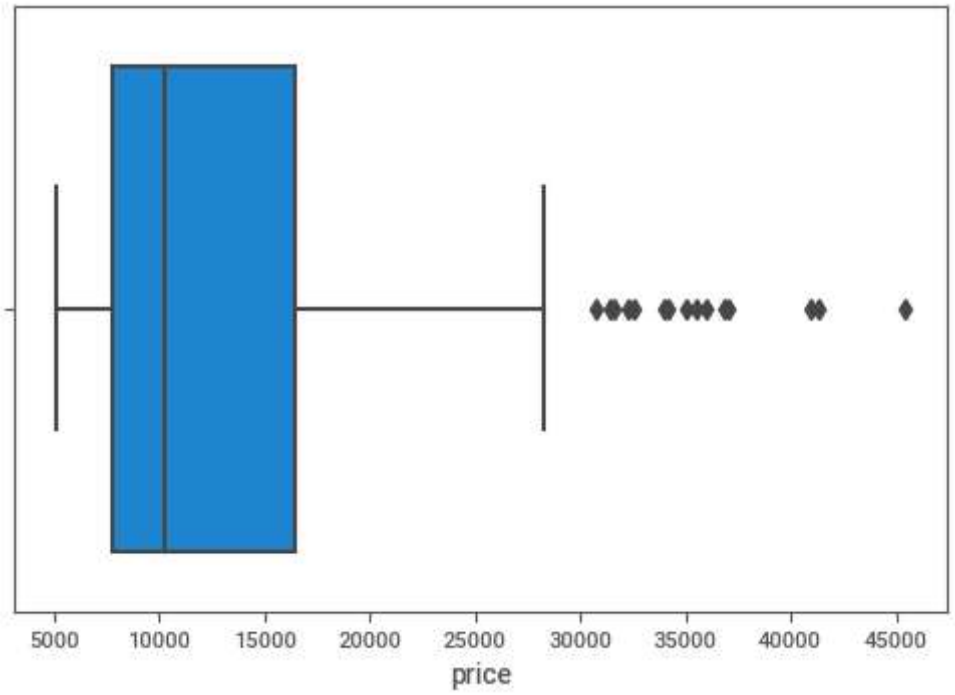
In [106…
```
sns.boxplot('carlength',data=X_features)
```

Out[106… <AxesSubplot:xlabel='carlength'>



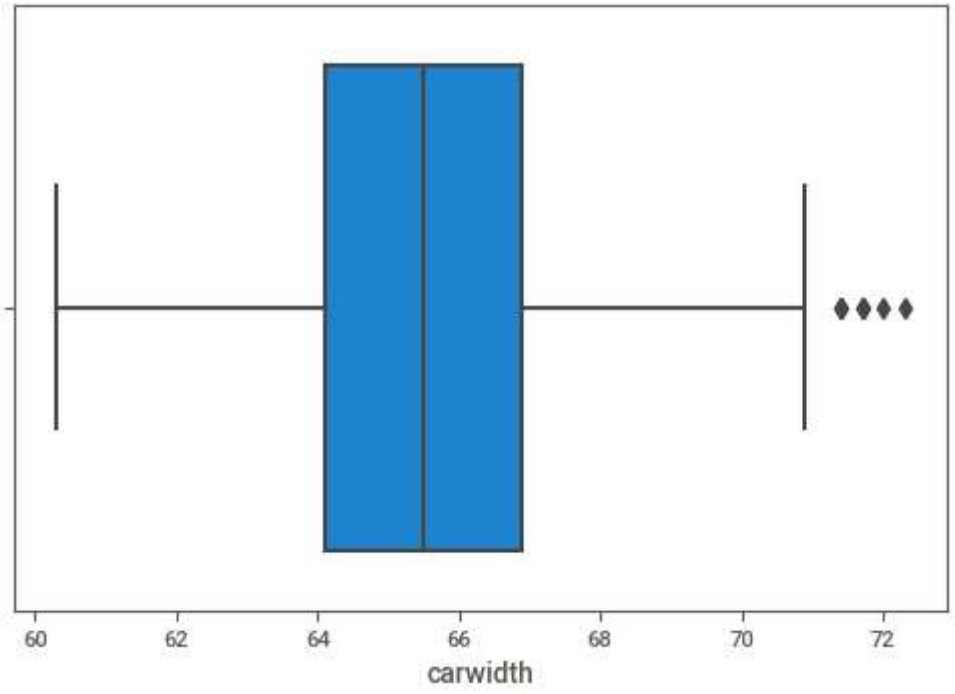
In [107…
```
sns.boxplot(y)
```

Out[107… <AxesSubplot:xlabel='price'>



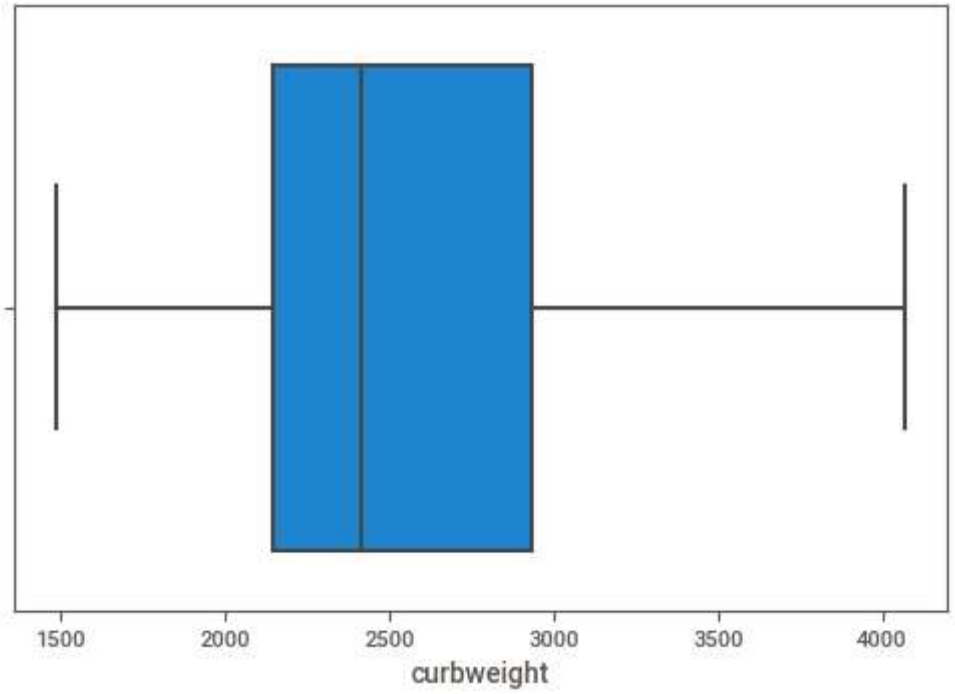
In [108…
```
sns.boxplot('carwidth',data=X_features)
```

Out[108… <AxesSubplot:xlabel='carwidth'>



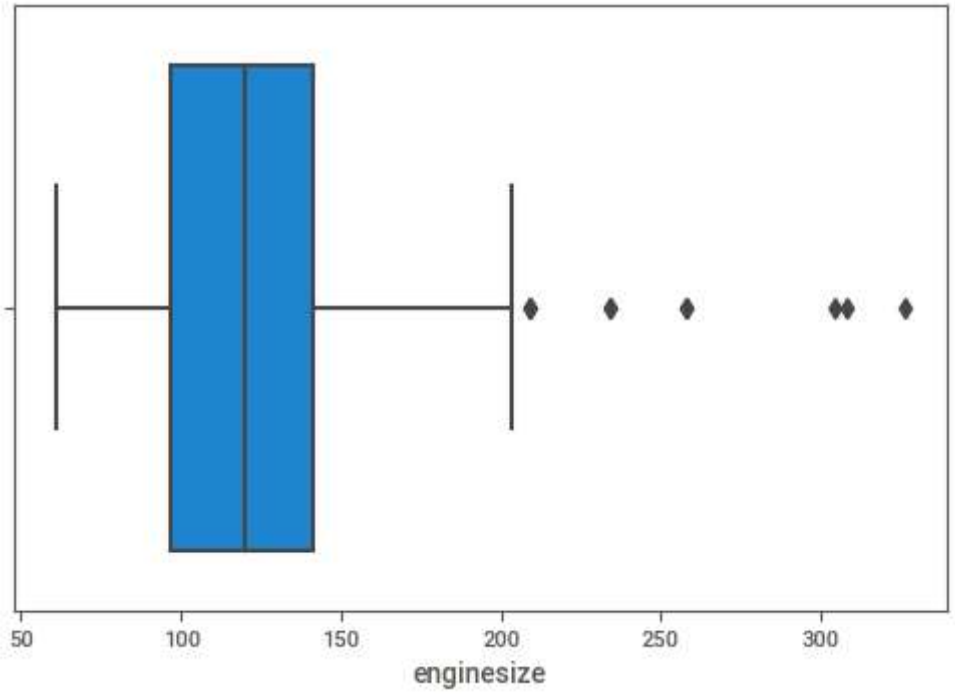
In [109…
```
sns.boxplot('curbweight',data=X_features)
```

Out[109… <AxesSubplot:xlabel='curbweight'>



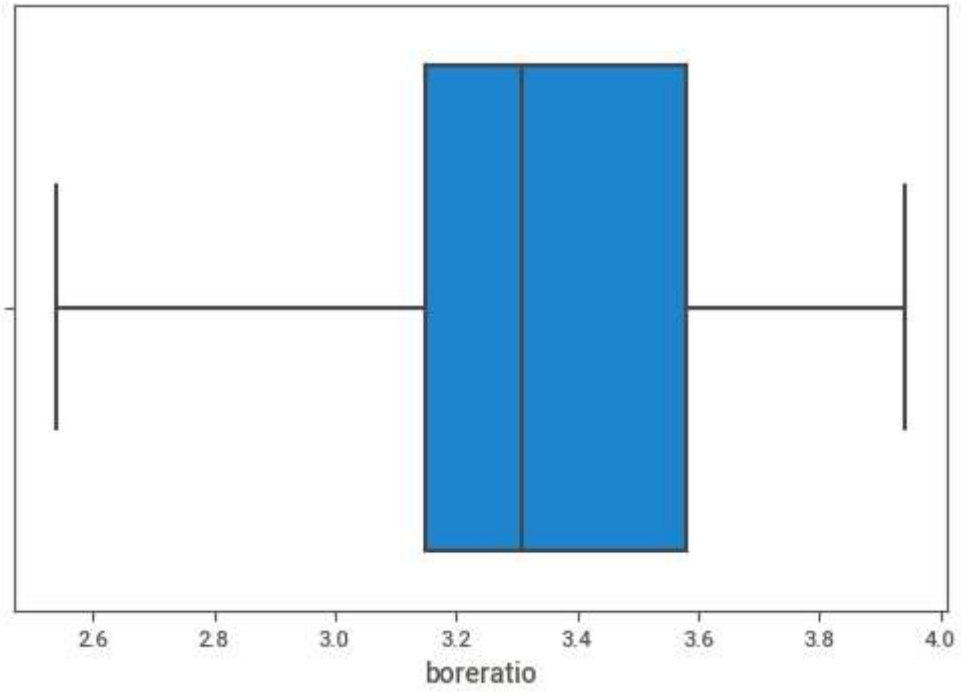
In [110…
```
sns.boxplot('enginesize',data=X_features)
```

Out[110… <AxesSubplot:xlabel='enginesize'>



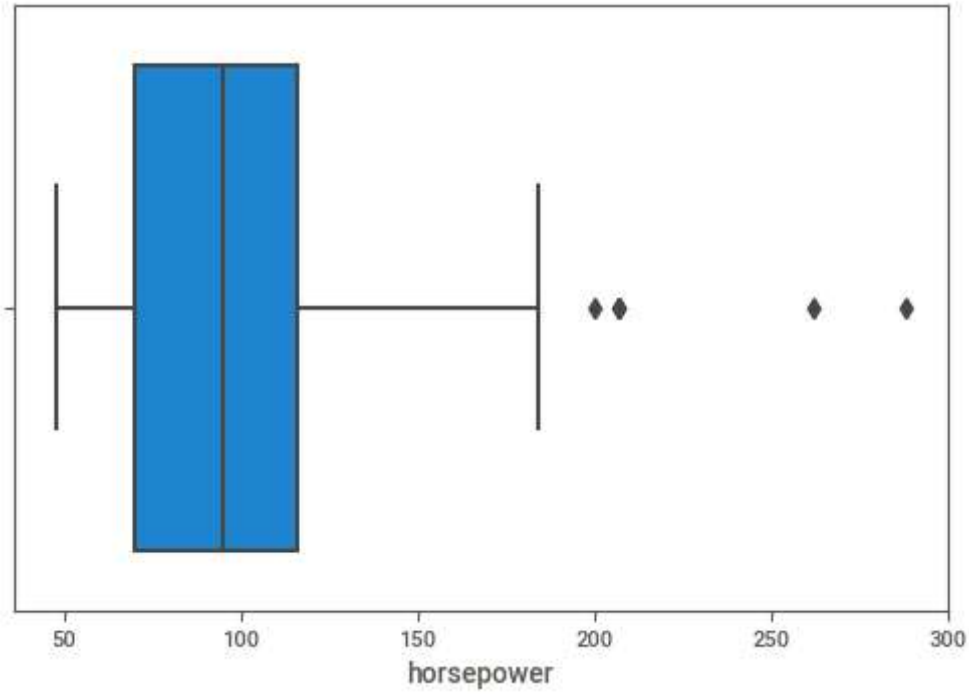
In [111…
```
sns.boxplot('boreratio',data=X_features)
```

Out[111…
```
<AxesSubplot:xlabel='boreratio'>
```



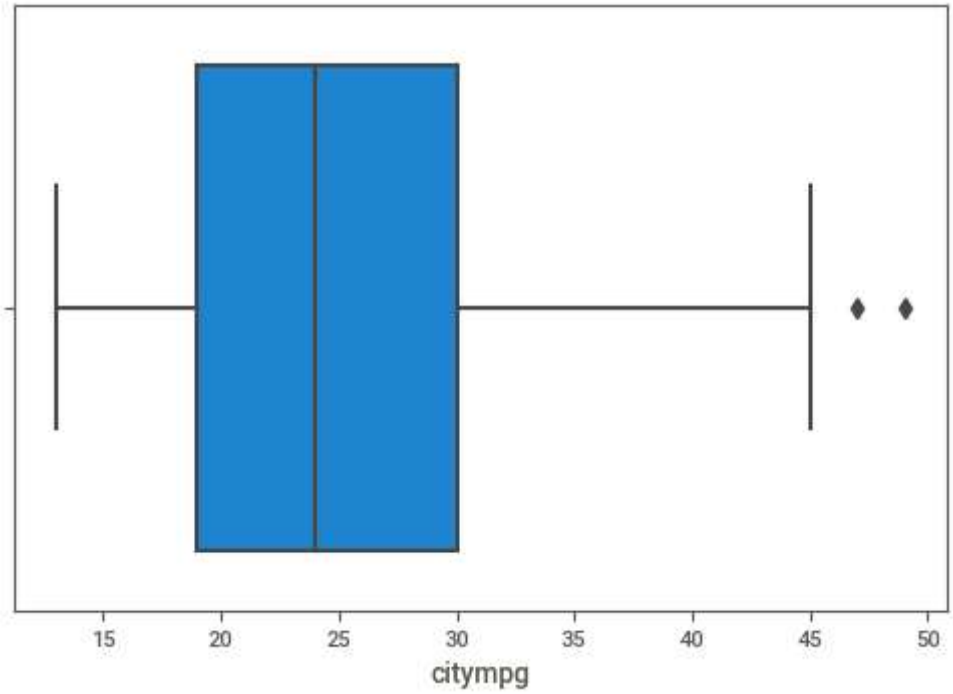
In [112…
```
sns.boxplot('horsepower',data=X_features)
```

Out[112…
```
<AxesSubplot:xlabel='horsepower'>
```



In [113…
```
sns.boxplot('citympg',data=X_features)
```

Out[113…
```
<AxesSubplot:xlabel='citympg'>
```



In [ ]:

In [ ]:

In [ ]: