In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
```

In [4]:
```
cp=pd.read_csv('car_price.csv')
```

In [5]:
```
cp.head()
```

Out[5]:

| | car_ID | symboling | CarName | fueltype | aspiration | doornumber | carbody | drivewheel | enginelocation | wheelbase | ... | enginesize | fuelsystem | boreratio | st |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 3 | alfa-romero giulia | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | |
| **1** | 2 | 3 | alfa-romero stelvio | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | |
| **2** | 3 | 1 | alfa-romero Quadrifoglio | gas | std | two | hatchback | rwd | front | 94.5 | ... | 152 | mpfi | 2.68 | |
| **3** | 4 | 2 | audi 100 ls | gas | std | four | sedan | fwd | front | 99.8 | ... | 109 | mpfi | 3.19 | |
| **4** | 5 | 2 | audi 100ls | gas | std | four | sedan | 4wd | front | 99.4 | ... | 136 | mpfi | 3.19 | |

5 rows × 26 columns

In [6]:
```
cp.shape
```

Out[6]:
```
(205, 26)
```

In [7]:
```
cp.isna().sum()
```

Out[7]:
```
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
```

In [8]:
```
cp.dtypes
```

Out[8]:
```
car_ID              int64
symboling           int64
CarName            object
fueltype           object
aspiration         object
doornumber         object
carbody            object
drivewheel         object
enginelocation     object
wheelbase         float64
carlength         float64
carwidth          float64
carheight         float64
curbweight          int64
enginetype         object
cylindernumber     object
enginesize          int64
fuelsystem         object
boreratio         float64
stroke            float64
compressionratio  float64
horsepower          int64
peakrpm             int64
citympg             int64
highwaympg          int64
price             float64
dtype: object
```

In [9]:
```
cp['CarName'].value_counts()
```

Out[9]:
```
toyota corona           6
toyota corolla          6
peugeot 504             6
subaru dl               4
mitsubishi mirage g4    3
                       ..
mazda glc 4             1
mazda rx2 coupe         1
maxda glc deluxe        1
maxda rx3               1
volvo 246               1
Name: CarName, Length: 147, dtype: int64
```

In [10]:
```
cp['aspiration'].value_counts()
```

Out[10]:
```
std      168
turbo     37
Name: aspiration, dtype: int64
```

In [11]:
```
cp.fueltype.value_counts()
```

Out[11]:
```
gas       185
diesel     20
Name: fueltype, dtype: int64
```

In [12]:
```
cp.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 [13]:
```
cp['cylindernumber'].value_counts()
```

Out[13]:
```
four      159
six        24
five       11
eight       5
two         4
three       1
twelve      1
Name: cylindernumber, dtype: int64
```

In [14]:
```
cp['enginelocation'].value_counts()
```

Out[14]:
```
front    202
rear       3
Name: enginelocation, dtype: int64
```

In [15]:
```
cp["fueltype"].value_counts()
```

Out[15]:
```
gas       185
diesel     20
Name: fueltype, dtype: int64
```

In [16]:
```
from sklearn.preprocessing import LabelEncoder
```

In [17]:
```
le=LabelEncoder()
```

In [18]:
```
cp['CarName']=le.fit_transform(cp['CarName'])
cp['fueltype']=le.fit_transform(cp['fueltype'])
cp['aspiration']=le.fit_transform(cp['aspiration'])
cp['doornumber']=le.fit_transform(cp['doornumber'])
cp['carbody']=le.fit_transform(cp['carbody'])
cp['drivewheel']=le.fit_transform(cp['drivewheel'])
cp['enginelocation']=le.fit_transform(cp['enginelocation'])
cp['cylindernumber']=le.fit_transform(cp['cylindernumber'])
cp['fuelsystem']=le.fit_transform(cp['fuelsystem'])
cp.enginetype=le.fit_transform(cp.enginetype)
```

In [19]:
```
cp.head()
```

Out[19]:

| | car_ID | symboling | CarName | fueltype | aspiration | doornumber | carbody | drivewheel | enginelocation | wheelbase | ... | enginesize | fuelsystem | boreratio | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 3 | 2 | 1 | 0 | 1 | 0 | 2 | 0 | 88.6 | ... | 130 | 5 | 3.47 | 2.68 |
| **1** | 2 | 3 | 3 | 1 | 0 | 1 | 0 | 2 | 0 | 88.6 | ... | 130 | 5 | 3.47 | 2.68 |
| **2** | 3 | 1 | 1 | 1 | 0 | 1 | 2 | 2 | 0 | 94.5 | ... | 152 | 5 | 2.68 | 3.47 |
| **3** | 4 | 2 | 4 | 1 | 0 | 0 | 3 | 1 | 0 | 99.8 | ... | 109 | 5 | 3.19 | 3.40 |
| **4** | 5 | 2 | 5 | 1 | 0 | 0 | 3 | 0 | 0 | 99.4 | ... | 136 | 5 | 3.19 | 3.40 |

5 rows × 26 columns

In [20]:
```
cp['cylindernumber'].value_counts()
```

Out[20]:
```
2    159
3     24
1     11
0      5
6      4
4      1
5      1
Name: cylindernumber, dtype: int64
```

In [21]:
```
cp.drop('car_ID',axis=1)
```

Out[21]:

| | symboling | CarName | fueltype | aspiration | doornumber | carbody | drivewheel | enginelocation | wheelbase | carlength | ... | enginesize | fuelsystem | boreratio | s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 2 | 1 | 0 | 1 | 0 | 2 | 0 | 88.6 | 168.8 | ... | 130 | 5 | 3.47 | |
| **1** | 3 | 3 | 1 | 0 | 1 | 0 | 2 | 0 | 88.6 | 168.8 | ... | 130 | 5 | 3.47 | |
| **2** | 1 | 1 | 1 | 0 | 1 | 2 | 2 | 0 | 94.5 | 171.2 | ... | 152 | 5 | 2.68 | |
| **3** | 2 | 4 | 1 | 0 | 0 | 3 | 1 | 0 | 99.8 | 176.6 | ... | 109 | 5 | 3.19 | |
| **4** | 2 | 5 | 1 | 0 | 0 | 3 | 0 | 0 | 99.4 | 176.6 | ... | 136 | 5 | 3.19 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **200** | -1 | 139 | 1 | 0 | 0 | 3 | 2 | 0 | 109.1 | 188.8 | ... | 141 | 5 | 3.78 | |
| **201** | -1 | 138 | 1 | 1 | 0 | 3 | 2 | 0 | 109.1 | 188.8 | ... | 141 | 5 | 3.78 | |
| **202** | -1 | 140 | 1 | 0 | 0 | 3 | 2 | 0 | 109.1 | 188.8 | ... | 173 | 5 | 3.58 | |
| **203** | -1 | 142 | 0 | 1 | 0 | 3 | 2 | 0 | 109.1 | 188.8 | ... | 145 | 3 | 3.01 | |
| **204** | -1 | 143 | 1 | 1 | 0 | 3 | 2 | 0 | 109.1 | 188.8 | ... | 141 | 5 | 3.78 | |

205 rows × 25 columns

In [22]:
```
cp.dtypes
```

Out[22]:
```
car_ID              int64
symboling           int64
CarName             int32
fueltype            int32
aspiration          int32
doornumber          int32
carbody             int32
drivewheel          int32
enginelocation      int32
wheelbase         float64
carlength         float64
carwidth          float64
carheight         float64
curbweight          int64
enginetype          int32
cylindernumber      int32
enginesize          int64
fuelsystem          int32
boreratio         float64
stroke            float64
compressionratio  float64
horsepower          int64
peakrpm             int64
citympg             int64
highwaympg          int64
price             float64
dtype: object
```

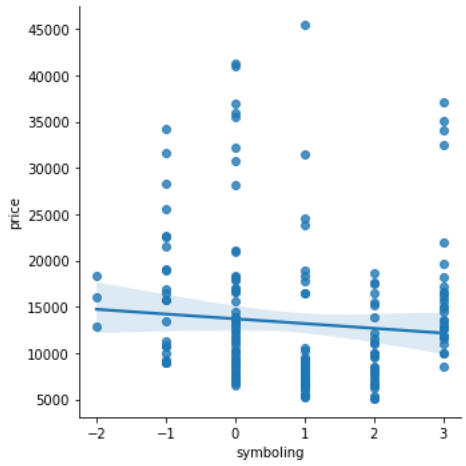
In [23]:
```
cp.enginetype.value_counts()
```

Out[23]:
```
3    148
4     15
5     13
0     12
2     12
6      4
1      1
Name: enginetype, dtype: int64
```

In [24]:
```
sns.lmplot('symboling','price',data=cp)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword ar
gs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[24]: <seaborn.axisgrid.FacetGrid at 0x2293f143130>




In [25]:
```
from sklearn.feature_selection import VarianceThreshold
from sklearn.model_selection import train_test_split
```

In [26]:
```
x_train,x_test,y_train,y_test=train_test_split(cp.drop(['price','car_ID','symboling'],axis=1),cp['price'],test_size=0.3,random_st
```

In [27]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(143, 23)
(62, 23)
```

In [28]:
```
var_thres=VarianceThreshold(threshold=0)
var_thres.fit(x_train)
```

Out[28]:
```
VarianceThreshold(threshold=0)
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [29]:
```
var_thres.get_support()
```

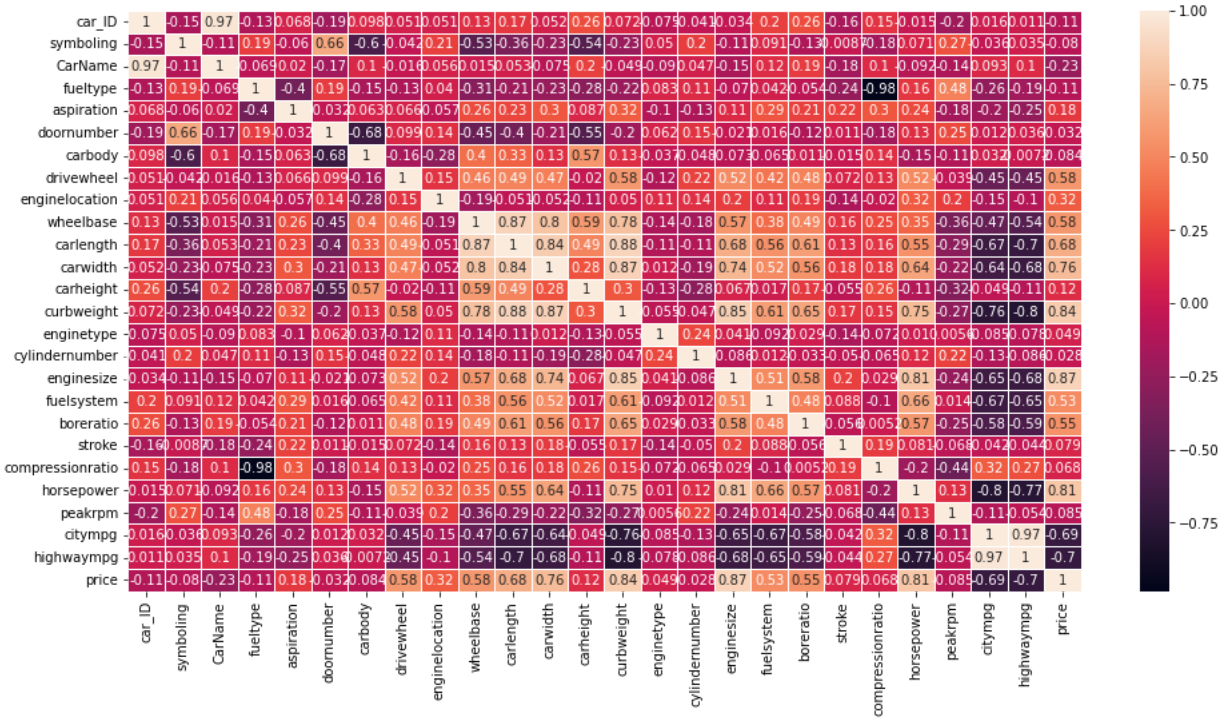
Out[29]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True])
```

In [30]:
```
### 2) we will check another method of feature selection
```

In [31]:
```
pc=cp.corr(method='pearson')
```

In [32]:
```
plt.figure(figsize=(16,8))
sns.heatmap(pc,xticklabels=pc.columns,yticklabels=pc.columns,annot=True,linewidth=0.5)
```

Out[32]: <AxesSubplot:>




In [33]:
```
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

In [34]:
```
corr=correlation(x_train, 0.8)
```

In [35]:
```
x_train=x_train.drop(corr,axis=1)
x_test=x_test.drop(corr,axis=1)
```

In [36]:
```
x_train.shape
```

Out[36]: (143, 16)

In [37]:
```
from sklearn.feature_selection import mutual_info_regression
# determine the mutual information
mutual_info = mutual_info_regression(x_train.fillna(0), y_train)
mutual_info
```

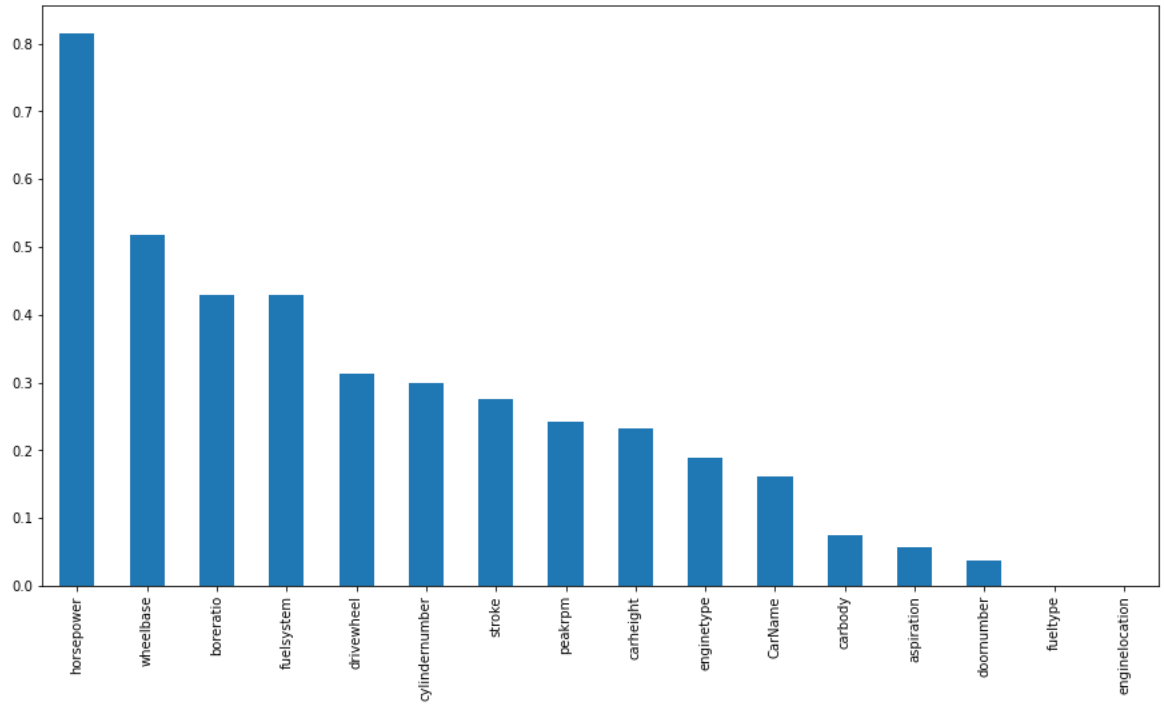
Out[37]:
```
array([0.16159854, 0.        , 0.05702623, 0.03780584, 0.07511562,
       0.31381441, 0.        , 0.51792621, 0.23128477, 0.18939549,
       0.29939881, 0.42862096, 0.42935158, 0.27511675, 0.81557144,
       0.24196995])
```

In [38]:
```
mutual_info=pd.Series(mutual_info)
mutual_info.index=x_train.columns
mutual_info.sort_values(ascending=False)
```

Out[38]:
```
horsepower        0.815571
wheelbase         0.517926
boreratio         0.429352
fuelsystem        0.428621
drivewheel        0.313814
cylindernumber    0.299399
stroke            0.275117
peakrpm           0.241970
carheight         0.231285
enginetype        0.189395
CarName           0.161599
carbody           0.075116
aspiration        0.057026
doornumber        0.037806
fueltype          0.000000
enginelocation    0.000000
dtype: float64
```

In [39]:
```
mutual_info.sort_values(ascending=False).plot.bar(figsize=(15,8))
```

Out[39]: <AxesSubplot:>




In [40]:
```
from sklearn.feature_selection import SelectPercentile
```

In [41]:
```
selected_top_columns=SelectPercentile(mutual_info_regression,percentile=30)
selected_top_columns.fit(x_train,y_train)
```

Out[41]:
```
SelectPercentile(percentile=30,
                 score_func=<function mutual_info_regression at 0x000002293FBB2D30>)
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [42]:
```
selected_top_columns.get_support()
```

Out[42]:
```
array([False, False, False, False, False,  True, False,  True, False,
       False, False,  True,  True, False,  True, False])
```

In [43]:
```
x_train.columns[selected_top_columns.get_support()]
```

Out[43]:
```
Index(['drivewheel', 'wheelbase', 'fuelsystem', 'boreratio', 'horsepower'], dtype='object')
```

In [44]:
```
x_train
```

Out[44]:

| | CarName | fueltype | aspiration | doornumber | carbody | drivewheel | enginelocation | wheelbase | carheight | enginetype | cylindernumber | fuelsystem | boreratio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **177** | 125 | 1 | 0 | 0 | 2 | 1 | 0 | 102.4 | 53.9 | 3 | 2 | 5 | 3.31 |
| **75** | 62 | 1 | 1 | 1 | 2 | 2 | 0 | 102.7 | 54.8 | 3 | 2 | 5 | 3.78 |
| **174** | 113 | 0 | 1 | 0 | 3 | 1 | 0 | 102.4 | 54.9 | 3 | 2 | 3 | 3.27 |
| **31** | 43 | 1 | 0 | 1 | 2 | 1 | 0 | 86.6 | 50.8 | 3 | 2 | 0 | 2.91 |
| **12** | 11 | 1 | 0 | 1 | 3 | 2 | 0 | 101.2 | 54.3 | 3 | 3 | 5 | 3.31 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **106** | 70 | 1 | 0 | 1 | 2 | 2 | 0 | 99.2 | 49.7 | 5 | 3 | 5 | 3.43 |
| **14** | 15 | 1 | 0 | 0 | 3 | 2 | 0 | 103.5 | 55.7 | 3 | 3 | 5 | 3.31 |
| **92** | 76 | 1 | 0 | 0 | 3 | 1 | 0 | 94.5 | 54.5 | 3 | 2 | 1 | 3.15 |
| **179** | 120 | 1 | 0 | 1 | 2 | 2 | 0 | 102.9 | 52.0 | 0 | 3 | 5 | 3.27 |
| **102** | 72 | 1 | 0 | 0 | 4 | 1 | 0 | 100.4 | 56.1 | 5 | 3 | 5 | 3.43 |

143 rows × 16 columns

In [45]:
```
columns=[['drivewheel', 'wheelbase', 'fuelsystem', 'boreratio', 'horsepower']]
```

In [46]:
```
X=cp[['drivewheel', 'wheelbase', 'fuelsystem', 'boreratio', 'horsepower']]
Y=cp.price
```

In [47]:
```
X.head()
```

Out[47]:

| | drivewheel | wheelbase | fuelsystem | boreratio | horsepower |
|---|---|---|---|---|---|
| **0** | 2 | 88.6 | 5 | 3.47 | 111 |
| **1** | 2 | 88.6 | 5 | 3.47 | 111 |
| **2** | 2 | 94.5 | 5 | 2.68 | 154 |
| **3** | 1 | 99.8 | 5 | 3.19 | 102 |
| **4** | 0 | 99.4 | 5 | 3.19 | 115 |

In [48]:
```
Y.head()
```

Out[48]:
```
0    13495.0
1    16500.0
2    16500.0
3    13950.0
4    17450.0
Name: price, dtype: float64
```

In [49]:
```
from sklearn.preprocessing import LabelEncoder
```

In [50]:
```
le=LabelEncoder()
```

In [51]:
```
X['drivewheel']=le.fit_transform(X['drivewheel'])
X['fuelsystem']=le.fit_transform(X['fuelsystem'])
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_4732\3394438188.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X['drivewheel']=le.fit_transform(X['drivewheel'])
C:\Users\Admin\AppData\Local\Temp\ipykernel_4732\3394438188.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  X['fuelsystem']=le.fit_transform(X['fuelsystem'])
```

In [52]:
```
X.head()
```

Out[52]:

| | drivewheel | wheelbase | fuelsystem | boreratio | horsepower |
|---|---|---|---|---|---|
| **0** | 2 | 88.6 | 5 | 3.47 | 111 |
| **1** | 2 | 88.6 | 5 | 3.47 | 111 |
| **2** | 2 | 94.5 | 5 | 2.68 | 154 |
| **3** | 1 | 99.8 | 5 | 3.19 | 102 |
| **4** | 0 | 99.4 | 5 | 3.19 | 115 |

In [53]:
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3,random_state=42)
```

In [54]:
```
x_test
```

Out[54]:

| | drivewheel | wheelbase | fuelsystem | boreratio | horsepower |
|---|---|---|---|---|---|
| **15** | 2 | 103.5 | 5 | 3.62 | 182 |
| **9** | 0 | 99.5 | 5 | 3.13 | 160 |
| **100** | 1 | 97.2 | 1 | 3.33 | 97 |
| **132** | 1 | 99.1 | 5 | 3.54 | 110 |
| **68** | 2 | 110.0 | 3 | 3.58 | 123 |
| **...** | ... | ... | ... | ... | ... |
| **56** | 2 | 95.3 | 2 | 3.33 | 101 |
| **128** | 2 | 89.5 | 5 | 3.74 | 207 |
| **76** | 1 | 93.7 | 1 | 2.97 | 68 |
| **144** | 0 | 97.0 | 1 | 3.62 | 82 |
| **104** | 2 | 91.3 | 5 | 3.43 | 160 |

62 rows × 5 columns

In [55]:
```
from sklearn.linear_model import LinearRegression
```

In [56]:
```
lr=LinearRegression()
```

In [57]:
```
lr.fit(x_train,y_train)
```

Out[57]:
```
LinearRegression()
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [58]:
```
pred_lr=lr.predict(x_test)
```

In [59]:
```
pred_lr
```

Out[59]:
```
array([26015.43055154, 19085.51580943, 11476.3290561 , 13324.04171249,
       20975.62044099,  6654.14154933,  5043.51996139,  7261.1897429 ,
       10307.80866859,  5197.7692757 , 13916.5371862 ,  8011.30255561,
       18180.29400535, 12263.34431234, 33213.22273153,  6210.46600759,
        1427.96892396, 16602.04610277,  9897.43025128, 10387.86603196,
       10350.11831929, 19534.83877609,  6894.31363947,  2212.88475262,
        6236.05712722, 26015.43055154, 15067.49235026, 17126.3210381 ,
        6654.14154933, 17126.3210381 , 20975.62044099,  6236.05712722,
        4980.00901461, 21787.84263022,  9751.84617395, 19654.67394525,
       11371.44383315, 12330.06136295,  6813.81589944, 16599.98545672,
        9312.31399473, 12082.79089299, 16069.30889997,  6210.46600759,
        6654.14154933,  9904.77346469,  6236.05712722,  6776.84812148,
       16331.78564286, 16909.85705876,  6813.81589944, 23013.50252561,
        4485.92727302, 12263.34431234,  6654.14154933, 13676.36509606,
       15067.49235026, 12330.06136295, 23760.35230003,  6236.05712722,
        8081.85700176, 18202.25851538])
```

In [60]:
```
import sklearn.metrics as metrics

print(np.sqrt(metrics.mean_squared_error(y_test,pred_lr)))
```

```
4058.5906745683906
```

In [61]:
```
print('R2 Value/Coefficient of Determination: {}'.format(lr.score(x_test, y_test)))
```

```
R2 Value/Coefficient of Determination: 0.7622529263614939
```

In [62]:
```
lin_score = lr.score(x_train, y_train)
print("R-squared:", lin_score)
```

```
R-squared: 0.7589742443135901
```

In [63]:
```
import statsmodels.api as sm
```

In [64]:
```
model = sm.OLS(y_train, x_train)
```

In [65]:
```
model=model.fit()
```

In [66]:
```
model.summary()
```

Out[66]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared (uncentered):** | 0.921 |
| **Model:** | OLS | **Adj. R-squared (uncentered):** | 0.918 |
| **Method:** | Least Squares | **F-statistic:** | 320.7 |
| **Date:** | Thu, 02 Feb 2023 | **Prob (F-statistic):** | 4.22e-74 |
| **Time:** | 22:38:35 | **Log-Likelihood:** | -1401.5 |
| **No. Observations:** | 143 | **AIC:** | 2813. |
| **Df Residuals:** | 138 | **BIC:** | 2828. |
| **Df Model:** | 5 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **drivewheel** | 2669.9771 | 853.837 | 3.127 | 0.002 | 981.683 | 4358.272 |
| **wheelbase** | 104.8079 | 50.454 | 2.077 | 0.040 | 5.044 | 204.572 |
| **fuelsystem** | 88.2013 | 239.181 | 0.369 | 0.713 | -384.732 | 561.135 |
| **boreratio** | -4685.6709 | 1602.763 | -2.923 | 0.004 | -7854.820 | -1516.522 |
| **horsepower** | 140.0880 | 13.478 | 10.394 | 0.000 | 113.438 | 166.738 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 42.283 | **Durbin-Watson:** | 2.147 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 97.937 |
| **Skew:** | 1.227 | **Prob(JB):** | 5.41e-22 |
| **Kurtosis:** | 6.227 | **Cond. No.** | 639. |

Notes:
[1] $R^2$ is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [67]:
```
cp=cp[["drivewheel","wheelbase","boreratio","horsepower","price"]]
```

In [68]:
```
X=cp.drop("price",axis=1)
Y=cp['price']
```

In [69]:
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3,random_state=42)
```

In [71]:
```
lr2=LinearRegression()
```

In [72]:
```
lr2.fit(x_train,y_train)
```

Out[72]:
```
LinearRegression()
```
**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [73]:
```
pred_lr2=lr2.predict(x_test)
```

In [74]:
```
pred_lr2
```

Out[74]:
```
array([25882.65795989, 19225.76055117, 11177.61713822, 13572.94552657,
       20722.39051504,  6493.98295728,  5231.17344511,  7086.81255221,
       10628.41954346,  5406.45027849, 14178.20164709,  7779.45769028,
       18273.24121504, 12575.27703533, 32947.66982682,  6076.00625218,
        1404.8034414 , 16888.04311674,  9638.73088065, 10707.16672   ,
       10088.31331735, 19617.0282626 ,  6730.22448691,  2006.28646403,
        6096.00750463, 25882.65795989, 15227.23472195, 17245.53800575,
        6493.98295728, 17245.53800575, 20722.39051504,  6096.00750463,
        4910.87805485, 21809.25659327, 10259.08692769, 19423.06210207,
       11690.64428572, 12118.73020299,  6657.02639488, 16885.37456036,
        9056.17040962, 12072.30703995, 16258.98652551,  6076.00625218,
        6493.98295728,  9508.80766231,  6096.00750463,  6577.65514628,
       16194.9411726 , 16882.70600399,  6657.02639488, 22988.37991904,
        4702.38920308, 12575.27703533,  6493.98295728, 13941.96011746,
       15227.23472195, 12118.73020299, 23613.8519542 ,  6096.00750463,
        7891.04221229, 18247.32601642])
```

In [75]:
```
print('R2 Value/Coefficient of Determination: {}'.format(lr2.score(x_test, y_test)))
```

```
R2 Value/Coefficient of Determination: 0.7528771956101769
```

In [78]:
```
model1 = sm.OLS(y_train, x_train)
```

In [79]:
```
model1=model1.fit()
```

In [80]:
```
model1.summary()
```

Out[80]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared (uncentered):** | 0.921 |
| **Model:** | OLS | **Adj. R-squared (uncentered):** | 0.918 |
| **Method:** | Least Squares | **F-statistic:** | 403.3 |
| **Date:** | Thu, 02 Feb 2023 | **Prob (F-statistic):** | 2.11e-75 |
| **Time:** | 22:48:37 | **Log-Likelihood:** | -1401.6 |
| **No. Observations:** | 143 | **AIC:** | 2811. |
| **Df Residuals:** | 139 | **BIC:** | 2823. |
| **Df Model:** | 4 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **drivewheel** | 2725.2062 | 837.982 | 3.252 | 0.001 | 1068.366 | 4382.046 |
| **wheelbase** | 105.1300 | 50.290 | 2.090 | 0.038 | 5.698 | 204.562 |
| **boreratio** | -4703.2892 | 1597.064 | -2.945 | 0.004 | -7860.968 | -1545.610 |
| **horsepower** | 142.4419 | 11.834 | 12.037 | 0.000 | 119.045 | 165.839 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 41.547 | **Durbin-Watson:** | 2.152 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 93.942 |
| **Skew:** | 1.217 | **Prob(JB):** | 3.99e-21 |
| **Kurtosis:** | 6.138 | **Cond. No.** | 639. |

Notes:
[1] $R^2$ is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```

```