Name : SINU S MARIAM

Designation : Data Science Intern

Organization : OASIS INFOBYTE

Batch : FEBRUARY P-2

#Task 3 - CAR PRICE PREDICTION WITH MACHINE LEARNING

Problem Statement:

- Analyse how various factors affect the price of the car
- Use Machine Learning Techniques for Predicting the car price using Python Programming

In [1]:

```
#importing necessary libraries
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
#importing libraries for visualisation
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
```

In [2]:

```
#importing Data
data_frame = pd.read_csv('C:/Users/sinun/OneDrive/Documents/oasis infobyte/car_price_prediction/CarPrice.csv')
```

Performing descriptive analysis. Understand the variables and their corresponding values.

In [3]:

```
# Understanding the dimensions of data
data_frame.shape
```

Out[3]:

```
(205, 26)
```

In [4]:

```python
# Understanding the Data Variables
data_frame.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 [5]:

```python
#Identify columns in Dataset
data_frame.columns
```

Out[5]:

```
Index(['car_ID', 'symboling', 'CarName', '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 [6]:

```
# Show the top 5 Rows of data
data_frame.head()
```

Out[6]:

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

5 rows × 26 columns

In [7]:

```
# Performing Descriptive Analysis
data_frame.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
car_ID	205.0	103.000000	59.322565	1.00	52.00	103.00	154.00	205.00
symboling	205.0	0.834146	1.245307	-2.00	0.00	1.00	2.00	3.00
wheelbase	205.0	98.756585	6.021776	86.60	94.50	97.00	102.40	120.90
carlength	205.0	174.049268	12.337289	141.10	166.30	173.20	183.10	208.10
carwidth	205.0	65.907805	2.145204	60.30	64.10	65.50	66.90	72.30
carheight	205.0	53.724878	2.443522	47.80	52.00	54.10	55.50	59.80
curbweight	205.0	2555.565854	520.680204	1488.00	2145.00	2414.00	2935.00	4066.00

	count	mean	std	min	25%	50%	75%	max
enginesize	205.0	126.907317	41.642693	61.00	97.00	120.00	141.00	326.00
boreratio	205.0	3.329756	0.270844	2.54	3.15	3.31	3.58	3.94
stroke	205.0	3.255415	0.313597	2.07	3.11	3.29	3.41	4.17
compressionratio	205.0	10.142537	3.972040	7.00	8.60	9.00	9.40	23.00
horsepower	205.0	104.117073	39.544167	48.00	70.00	95.00	116.00	288.00
peakrpm	205.0	5125.121951	476.985643	4150.00	4800.00	5200.00	5500.00	6600.00
citympg	205.0	25.219512	6.542142	13.00	19.00	24.00	30.00	49.00
highwaympg	205.0	30.751220	6.886443	16.00	25.00	30.00	34.00	54.00
price	205.0	13276.710571	7988.852332	5118.00	7788.00	10295.00	16503.00	45400.00

In [8]:

```
# Checking for null values
data_frame.isnull().sum()
```

Out[8]:

```
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 [9]:
```
#Dropping unwanted Columns from data
data_frame.drop(columns=['car_ID'], inplace=True )
```

In [10]:
```
# Identifing Categorical Columns in Dataset
data_frame_cat=data_frame.select_dtypes(exclude=['float64','int64'])
data_frame_cat.columns
```

Out[10]:
```
Index(['CarName', 'fueltype', 'aspiration', 'doornumber', 'carbody',
       'drivewheel', 'enginelocation', 'enginetype', 'cylindernumber',
       'fuelsystem'],
      dtype='object')
```

In [11]:
```
# Identifing Numerical Columns in Dataset
data_frame_num=data_frame.select_dtypes(include=['float64','int64'])
data_frame_num.columns
```

Out[11]:
```
Index(['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight',
       'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price'],
      dtype='object')
```

In [12]:
```
# Identifing Unique values in each Categorical column
cat_cols=['CarName','fueltype', 'aspiration', 'doornumber', 'carbody',
       'drivewheel', 'enginelocation', 'enginetype', 'cylindernumber',
       'fuelsystem']
def num_count():
    for col in cat_cols:
        print('Name of the variable :', col)
        print(data_frame[col].value_counts(), '\n\n')
num_count()
```

```
Name of the variable : CarName
toyota corona           6
toyota corolla          6
peugeot 504             6
```

```
subaru dl                  4
toyota mark ii             3
                          ..
dodge dart custom          1
buick opel isuzu deluxe    1
subaru tribeca             1
volkswagen super beetle    1
buick century              1
Name: CarName, Length: 147, dtype: int64


Name of the variable : fueltype
gas       185
diesel     20
Name: fueltype, dtype: int64


Name of the variable : aspiration
std      168
turbo     37
Name: aspiration, dtype: int64


Name of the variable : doornumber
four    115
two      90
Name: doornumber, dtype: int64


Name of the variable : carbody
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: carbody, dtype: int64


Name of the variable : drivewheel
fwd    120
rwd     76
4wd      9
Name: drivewheel, dtype: int64


Name of the variable : enginelocation
front    202
```

```
rear        3
Name: enginelocation, dtype: int64


Name of the variable : enginetype
ohc      148
ohcf      15
ohcv      13
dohc      12
l         12
rotor      4
dohcv      1
Name: enginetype, dtype: int64


Name of the variable : cylindernumber
four      159
six        24
five       11
eight       5
two         4
twelve      1
three       1
Name: cylindernumber, dtype: int64


Name of the variable : fuelsystem
mpfi    94
2bbl    66
idi     20
1bbl    11
spdi     9
4bbl     3
spfi     1
mfi      1
Name: fuelsystem, dtype: int64
```

Data Visualization

*** Data Visualization helps to show how the differnt factors affect the Price Variable**

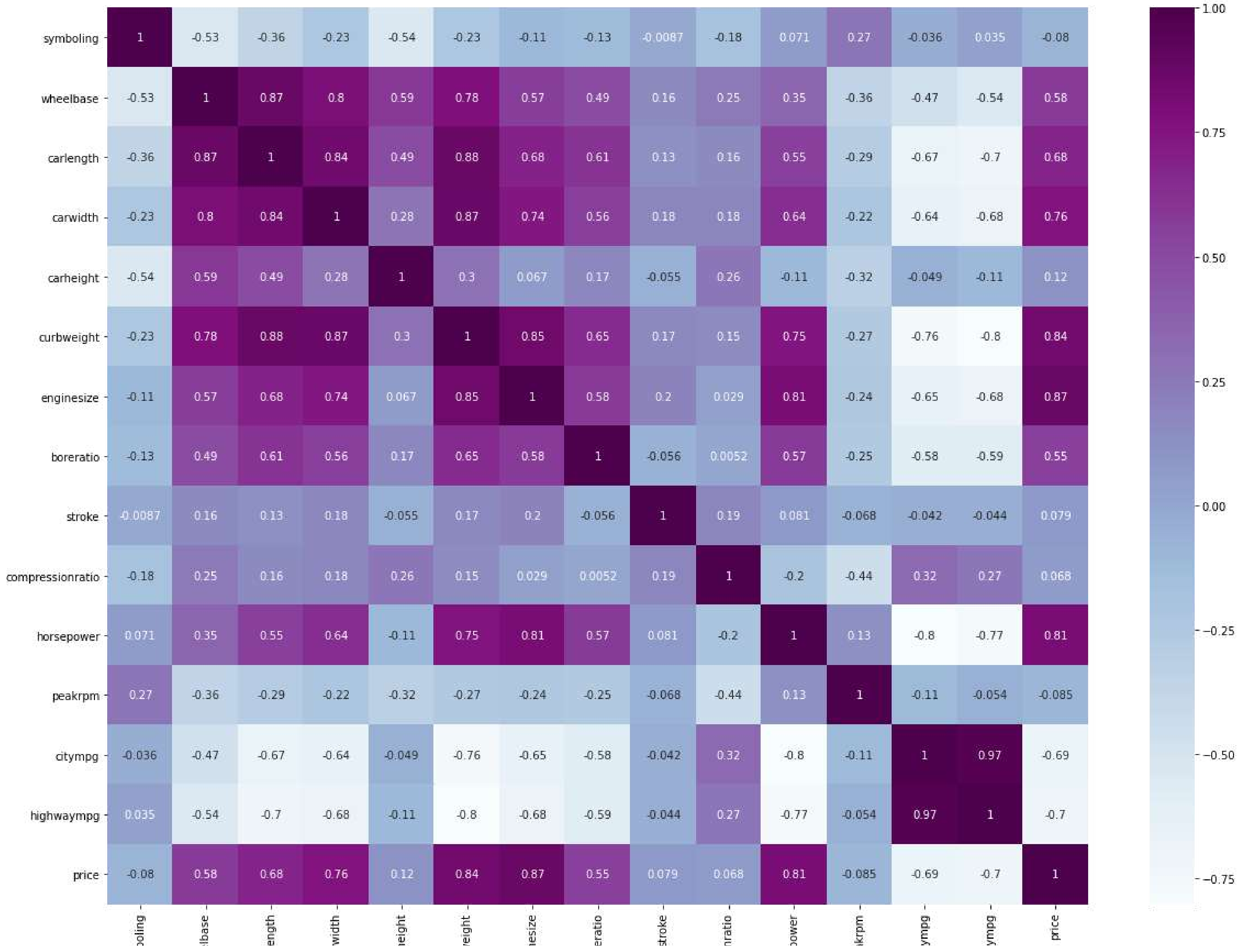
Heat Map

In [13]:

```
# find correlation between variables in data set for plotting heatmap
df_corr=data_frame.corr()
```

In [14]:

```
plt.figure(figsize=(20,15))
sns.heatmap(df_corr,annot=True,cmap="BuPu")
plt.show()
```


symboling 1 -0.53 -0.36 -0.23 -0.54 -0.23 -0.11 -0.13 -0.0087 -0.18 0.071 0.27 -0.036 0.035 -0.08
wheelbase -0.53 1 0.87 0.8 0.59 0.78 0.57 0.49 0.16 0.25 0.35 -0.36 -0.47 -0.54 0.58
carlength -0.36 0.87 1 0.84 0.49 0.88 0.68 0.61 0.13 0.16 0.55 -0.29 -0.67 -0.7 0.68
carwidth -0.23 0.8 0.84 1 0.28 0.87 0.74 0.56 0.18 0.18 0.64 -0.22 -0.64 -0.68 0.76
carheight -0.54 0.59 0.49 0.28 1 0.3 0.067 0.17 -0.055 0.26 -0.11 -0.32 -0.049 -0.11 0.12
curbweight -0.23 0.78 0.88 0.87 0.3 1 0.85 0.65 0.17 0.15 0.75 -0.27 -0.76 -0.8 0.84
enginesize -0.11 0.57 0.68 0.74 0.067 0.85 1 0.58 0.2 0.029 0.81 -0.24 -0.65 -0.68 0.87
boreratio -0.13 0.49 0.61 0.56 0.17 0.65 0.58 1 -0.056 0.0052 0.57 -0.25 -0.58 -0.59 0.55
stroke -0.0087 0.16 0.13 0.18 -0.055 0.17 0.2 -0.056 1 0.19 0.081 -0.068 -0.042 -0.044 0.079
compressionratio -0.18 0.25 0.16 0.18 0.26 0.15 0.029 0.0052 0.19 1 -0.2 -0.44 0.32 0.27 0.068
horsepower 0.071 0.35 0.55 0.64 -0.11 0.75 0.81 0.57 0.081 -0.2 1 0.13 -0.8 -0.77 0.81
peakrpm 0.27 -0.36 -0.29 -0.22 -0.32 -0.27 -0.24 -0.25 -0.068 -0.44 0.13 1 -0.11 -0.054 -0.085
citympg -0.036 -0.47 -0.67 -0.64 -0.049 -0.76 -0.65 -0.58 -0.042 0.32 -0.8 -0.11 1 0.97 -0.69
highwaympg 0.035 -0.54 -0.7 -0.68 -0.11 -0.8 -0.68 -0.59 -0.044 0.27 -0.77 -0.054 0.97 1 -0.7
price -0.08 0.58 0.68 0.76 0.12 0.84 0.87 0.55 0.079 0.068 0.81 -0.085 -0.69 -0.7 1
width
stroke
price
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

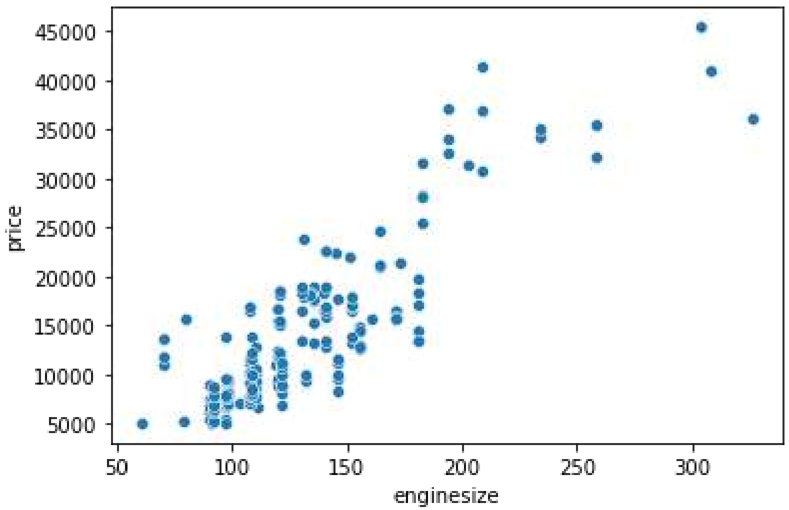
* Variables Enginesize,curbweight,horsepower have high correlation values (above 0.8) with the target Price variable

* Factors such as carwidth,carlength,highwaympg and citympg are also having good correlation values (above 0.68) with the target Price variable

In [15]:

```
#Scatter plot is used find the how various factors affect the price of a car
#SCATTER PLOT ENGINE SIZE vs Price
plt.figure(figsize=(6,4))
sns.scatterplot(data=data_frame,x=data_frame['enginesize'],y=data_frame['price'])
```

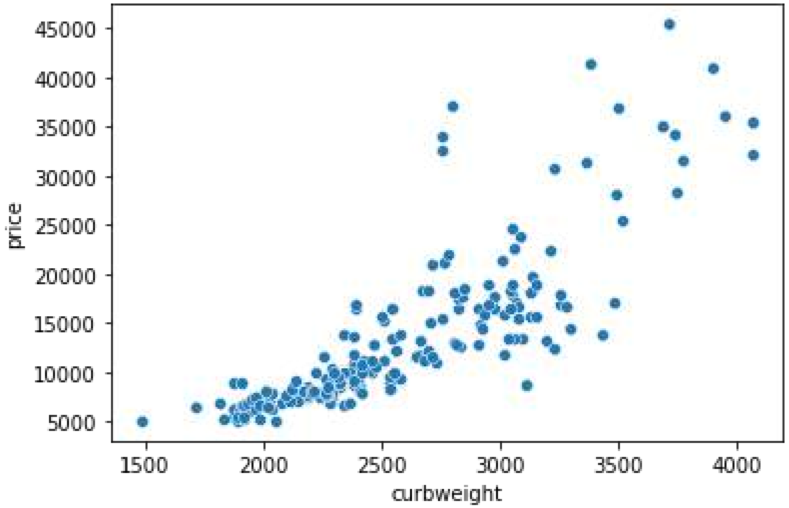
Out[15]: <AxesSubplot:xlabel='enginesize', ylabel='price'>



In [16]:

```
#SCATTER PLOT CURBWEIGHT vs Price
plt.figure(figsize=(6,4))
sns.scatterplot(data=data_frame,x=data_frame['curbweight'],y=data_frame['price'])
```

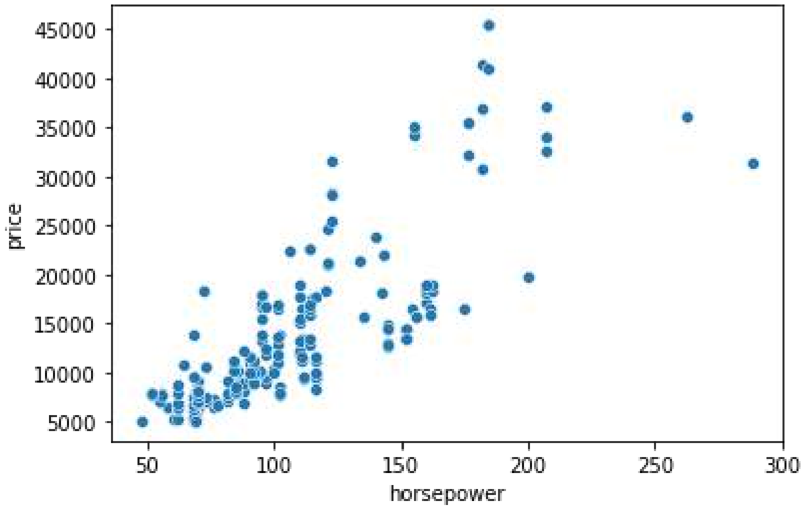
Out[16]: <AxesSubplot:xlabel='curbweight', ylabel='price'>



In [17]:

```
#SCATTER PLOT HORSEPOWER vs Price
plt.figure(figsize=(6,4))
sns.scatterplot(data=data_frame,x=data_frame['horsepower'],y=data_frame['price'])
```

Out[17]: <AxesSubplot:xlabel='horsepower', ylabel='price'>



*** It is seen that Enginesize, curbweight, horsepowe mostly follows a linear relationship with the Price variable.**

In [18]:

```
#One Hot encoding for categorical variables for labelling the categorical columns
data_frame= pd.get_dummies(data_frame, columns = ['fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel', 'cylindernumber
data_frame.head(5)
```

Out[18]:

	symboling	CarName	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	...	enginetype_ohcv	enginetype_rotor	f
0	3	alfa-romero giulia	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	...	0	0	
1	3	alfa-romero stelvio	88.6	168.8	64.1	48.8	2548	130	3.47	2.68	...	0	0	
2	1	alfa-romero Quadrifoglio	94.5	171.2	65.5	52.4	2823	152	2.68	3.47	...	1	0	
3	2	audi 100 ls	99.8	176.6	66.2	54.3	2337	109	3.19	3.40	...	0	0	
4	2	audi 100ls	99.4	176.6	66.4	54.3	2824	136	3.19	3.40	...	0	0	

5 rows × 54 columns

Building the Model

In [19]:

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

In [20]:

```
#First step in building the model is to identify the Feature(Input) variables and Target (Output) variable
features = data_frame.drop(['CarName','price'], axis=1)
target = data_frame['price']
```

* Splitting data for training and testing the model

In [21]:

```
# Splitting data for training the model and testing the model
# train size taken as 0.8
X_train, X_test, y_train, y_test = train_test_split(features, target, train_size = .8)
# Dimensions of Train and Test Data sets
print('Train set of features: ', X_train.shape)
print('Test set of features: ', X_test.shape)
```

```
print('Target for train: ', y_train.shape)
print('Target for test: ', y_test.shape)
```

```
Train set of features:  (164, 52)
Test set of features:  (41, 52)
Target for train:  (164,)
Target for test:  (41,)
```

Learn the model on train data

In [22]:

```
from sklearn.linear_model import LinearRegression
```

In [23]:

```
# Linear Regression Model ( a Supervised Machine learning Algorithm)
# LR models impose a linear function between predictor and response variables
my_model = LinearRegression()
```

In [24]:

```
# Fitting the model in train data set ie the Linear Regression Model learned from the on Train Data
my_model.fit(X_train, y_train)
```

Out[24]:

```
LinearRegression()
```

Predicting the Car Price

In [25]:

```
# Predicting the car price from Feature Test values
y_pred = my_model.predict(X_test)
y_pred
```

Out[25]:

```
array([35073.37736929, 12257.88245737, 12439.60583596, 23099.8618908 ,
       18621.58849834,  7692.81971512, 33275.72758724,  9028.50543349,
       21258.89757212,  6097.87963715, 10964.52044721,  8039.46757434,
       11142.40842091, 17854.62293629, 34028.        , 32145.42950074,
       17051.70802151,  7774.13729871, 11184.55513878,  5350.94624111,
       14757.19533186, 12307.01072466, 17998.08282027, 15996.66264422,
        6365.59455277,  6147.9396388 ,  6881.78557215, 20140.76421575,
        9122.84043963,  5964.63948908,  9889.42308737, 11390.71898048,
       36091.8682374 ,  8723.7265066 , 26755.7166857 , 10396.98003988,
        5594.20784798, 21598.78924043,  6437.64017054,  5560.21737511,
        5983.38165651])
```

Test the model

In [26]:
```
from sklearn.metrics import mean_squared_error
```

Mean Squared Error

In [27]:
```
# Compare the predicted values with the true values
mean_squared_error(y_pred, y_test)
```

Out[27]:
```
7096542.344329189
```

Coefficient of Determination or R Squared Value (r2)

In [28]:
```
from sklearn.metrics import r2_score
```

In [29]:
```
# find Coefficient of Determination or R Squared Value (r2)
r2_score(y_test,y_pred)
```

Out[29]:
```
0.9118853055649259
```