Project: Determination of the price of Automobile

Determination of the price of the automobile by considering various parameters by using the appropriate ML Algorithm

Name: Kevin Prince

USN:1BY18EC090

Email:kevinprinc06@gmail.com

Company Name: innovation. creation

Instructor's Name: Abhishek C

Acknowledgement

Firstly, I would like to acknowledge Mr. Abhishek C for guiding me whenever I had a roadblock at any position, without any kind of hesitation he would text me back for the help, so I am thankful to him. I would like to thank TakeItEasy Engineering and Innovation creations for providing me with such an opportunity which has benefited me and a lot of students.

I have done this project with my heart and utmost sincerity to complete the project assigned to me. By doing this project it helped me doing a lot of research and enhancing my knowledge in this subject.

Secondly, I would like to thank my friends without whom I wouldn't able to complete my project, through their support and guidance I was able to complete this project. Once again, I would like to thank everyone whom I have not mentioned.

Kevin Prince

innovation.creation

Project: Determination of the price of Automobile

Abstract

In the following project, my task is to predict the price of the automobile through different parameters like highway-mileage, city-mpg, horsepower, num-of-cylinder, and other factors.

This data set consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) it assigned insurance risk rating, (c) its normalized losses in use as compared to other cars. The second rating corresponds to the degree to which the auto is riskier than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is riskier (or less), this symbol is adjusted by moving it up (or down) the scale. Actuaries call this process "symbolling". A value of +3 indicates that the auto is risky, - 3 that it is probably pretty safe. The third factor is the relative average loss payment per insured vehicle year. This value is normalized for all autos within a particular size classification (two-door small, station wagons, sports/specialty, etc...), and represents the average loss per car per year.

The main objective is to prove that there is a direct relation between the price and other parameters say for example symbolling. Symbolising is nothing but the scaling factor of the car as discussed earlier.

It is seen that with the increase in price the symbolling value decreases as expected, but neglecting all the anomalies found in the graph, it is to be true. Similar to this it is observed that parameters like horsepower, curb-weight, and weight of the engine is directly correlated with the price of the automobile.

By cleaning the data, training and testing these data with various algorithms to see which algorithm has got the highest accuracy, and so on.

About the Companies

ICS is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At ICS, we believe that service and quality is the key to success.

We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

- Development - We develop responsive, functional and super-fast websites. We keep User Experience in mind while creating websites. A website should load quickly and should be accessible even on a small view-port and slow internet connection.
- Mobile Application - We offer a wide range of professional Android, iOS & Hybrid app development services for our global clients, from a start-up to a large enterprise.
- Design - We offer professional Graphic design, Brochure design & Logo design. We are experts in crafting visual content to convey the right message to the customers.
- Consultancy - We are here to provide you with expert advice on your design and development requirement.
- Videos - We create a polished professional video that impresses your audience.

innovation.creation

Project: Determination of the price of Automobile

Index

Introduction

Our main objective is to predict the price of the automobile using various parameters. As discussed earlier This data set consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) it assigned insurance risk rating, (c) its normalized losses in use as compared to other cars. The second rating corresponds to the degree to which the auto is riskier than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is riskier (or less), this symbol is adjusted by moving it up (or down) the scale. Actuaries call this process "symbolling". A value of +3 indicates that the auto is risky, - 3 that it is probably pretty safe. The third factor is the relative average loss payment per insured vehicle year. This value is normalized for all autos within a particular size classification (two-door small, station wagons, sports/specialty, etc...), and represents the average loss per car per year.

With these datasets which consist of 26 different parameters and with these parameters, my task is to determine the price using the appropriate algorithm which has got the highest accuracy. but before finding out the accuracy the data has to be trained and then tested. But in order to train these data, we need to have the appropriate values and it is observed that there are many unknowns in the database so it has to be cleaned with the appropriate value. By using various algorithms like linear regression, SVM regression method, decision tree (regression version), and many more.

By determining the correlation of the variables with respect to the price, each graph is being plotted. Also, to determine the mode of certain variables, I have used countplot and substitute the mode with the unknown, and a brief explanation will be given in further discussion.

Problem Statement and Objective

Given a set of data that consists of many parameters and with respect to this parameter, the price of the automobile has to be determined. Since we are asked to determine the price of the automobile it leads to linear regression since the price has to be determined using various parameters. Our objective is to find the appropriate algorithm of linear regression like stochastic, SVM, Decision Tree, Random forest, and many more.

Firstly, we need to clean the data given since there are many unknowns which are given to us, these values have to be replaced in order to proceed with further steps. Once the data is cleaned i.e., by putting the appropriate values wrt each variable.

Then the correlation is found out with respect to 'price' and the variable, and the parameter which is closer to the 1 is strong correlate and their plots are being plotted.

Next, we convert all the object types to numerical using certain functions in order to train and test these data sets. Before this step variables of an object type that are continuous are being converted to float, since these variables are not discrete in nature and the function cannot convert object datasets into other data types.

As discussed earlier there are many unknowns represented by '?', these values have to be cleaned. So by using NumPy library, we replace these values with Nan(Not A Number), which eases our life by reducing the complexity in the calculation and by using mean function these value, but note that each variable cannot be replaced with these mean values, for instance, a car cannot have 3.73 as door since the number of doors can be 2 or 4. Therefore the mode of these values has to be determined and can be replaced by these unknown.

Whereas other variables can be replaced by the mean of the variable, which gives more clarity to the relation.

Requirement Specification

Hardware:

- Processor: AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz
- RAM: 8.00GB
- System Type: 64-bit operating system, x64 based processor
- Edition: Windows 10 Home
- Laptop: Asus ZenBook 2020 edition
- Storage: 512 GB SSD

Software:

- Anaconda Prompt
- Jupyter Notebook

Libraries:

- NumPy
- Seaborn
- MatPlot
- Pandas
- Scikit

Exploratory Data analysis

- Firstly, all the libraries required for this project is being imported to the Jupyter notebook.
  #Importing all the required libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from statistics import mean
%matplotlib inline
```

- Now the data from the csv file hast to be imported in order to proceed further steps.

```
data=pd.read_csv("Automobile price data _Raw_.csv")
data=pd.DataFrame(data)
print("Data imported successfully")
```

Out: `Data imported successfully`

- In order to see whether the dataset has imported successfully, we check the head part of the data set.

  #showing the first 10 datasets of the data
  data.head(10)

innovation.creation

Project: Determination of the price of Automobile

Note: Output displayed in the following page

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0
5	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5
7	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3
9	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0

10 rows × 26 columns

horsepower	peak-rpm	city-mpg	highway-mpg	price
111	5000	21	27	13495
111	5000	21	27	16500
154	5000	19	26	16500
102	5500	24	30	13950
115	5500	18	22	17450
110	5500	19	25	15250
110	5500	19	25	17710
110	5500	19	25	18920
140	5500	17	20	23875
160	5500	16	22	?

Following is the first 10 data elements of the data set

Project: Determination of the price of Automobile

- This function is used to describe each datasets and type of values it holds, number of values each column consists of.

```
%to describe the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       205 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    object
 19  stroke             205 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object
 22  peak-rpm           205 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

- Since there any many unknown in the datasets which are denoted by '?'. Hence it is necessary to convert these values to Nan (Not a Number), which helps to determine the mean, mode etc for further steps without any error.
- Further we determine the no of unknown in each column

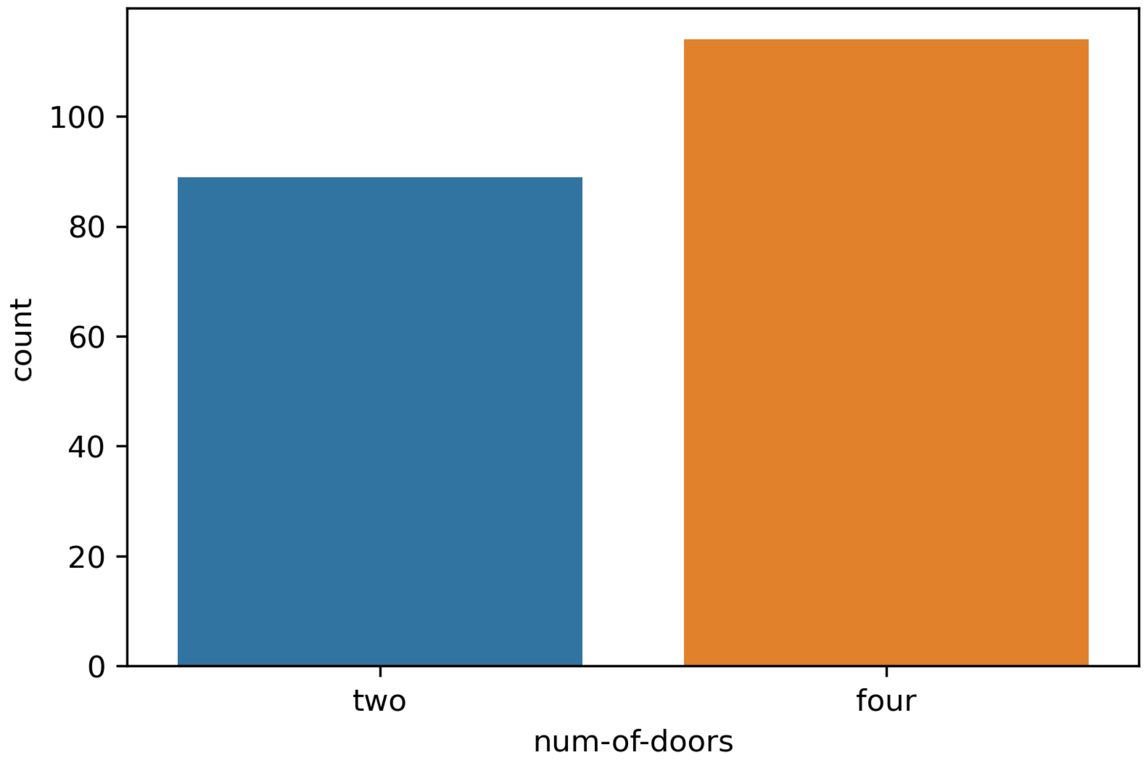
#This process is used to convert all the unknown('?') with Nan and converting it as float from obj

```
data.replace({'?':np.nan},inplace=True)   #used to replace all '?' with Nan
count_nan_in_data = data.isnull().sum()  #to count the no of unknown
print (count_nan_in_data)
```

```
symboling             0
normalized-losses    41
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 4
```

- It is observed that variables like normalized-losses, num-of-doors, bore, stroke, horsepower, peak-rpm and price consists of unknowns. Mean of these columns can be replaced with the unknown values, but in case of num-of-doors it cannot be done so, therefore mode of the num-of-doors is determined and is being replaced by the missing values.

```
#Countplot is being used to find the mode of the Num-of-doors
sns.countplot(x='num-of-doors',data=data)
```



It is observed that Automobile consisting of four doors has got the maximum count and ‘four’ is being replaced in the missing values of the column num-of-doors.

```
#It is observed that Cars having four doors is the mode.
```

```
data['num-of-doors'][data['num-of-doors']=='?']='four'
```

- As discussed earlier missing value of other column has to be replaced by the mean, in order to do so we need to convert the object variables into 'Float'

```
data[['normalized-losses','horsepower','peak-rpm','price','bore','stroke']]=data[['normalized-losses','horsepower','peak-rpm','price','bore','stroke']].astype(float)
```

- We create a list of the columns of the 'Object type' and its being converted to 'Float' using astype function.
- Now these values will be replaced by the mean of the individual column by using replace function.
- By using inplace=True, it will find the mean of individual column and replace it wherever it finds Nan permanently.

```
data['normalized-losses'].replace({np.nan:data['normalized-losses'].mean()},inplace=True)
data['horsepower'].replace({np.nan:data['horsepower'].mean()},inplace=True)
data['peak-rpm'].replace({np.nan:data['peak-rpm'].mean()},inplace=True)
data['price'].replace({np.nan:data['price'].mean()},inplace=True)
data['bore'].replace({np.nan:data['bore'].mean()},inplace=True)
data['stroke'].replace({np.nan:data['stroke'].mean()},inplace=True)
```

- By using info function, to check the data type of each induvial variable and making sure that they have converted its appropriate datatype.

```
#to check if the continuous object has successfully converted to float
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    float64
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   aspiration         205 non-null    object 
 5   num-of-doors       203 non-null    object 
 6   body-style         205 non-null    object 
 7   drive-wheels       205 non-null    object 
 8   engine-location    205 non-null    object 
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64  
 14  engine-type        205 non-null    object 
 15  num-of-cylinders   205 non-null    object 
 16  engine-size        205 non-null    int64  
 17  fuel-system        205 non-null    object 
 18  bore               205 non-null    float64
 19  stroke             205 non-null    float64
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    float64
 22  peak-rpm           205 non-null    float64
 23  city-mpg           205 non-null    int64  
 24  highway-mpg        205 non-null    int64  
 25  price              205 non-null    float64
dtypes: float64(11), int64(5), object(10)
memory usage: 41.8+ KB
```

- It is observed that the changes made in dataset are reflecting.

- Describing each individual variable using describe function.

```
data.describe()
```

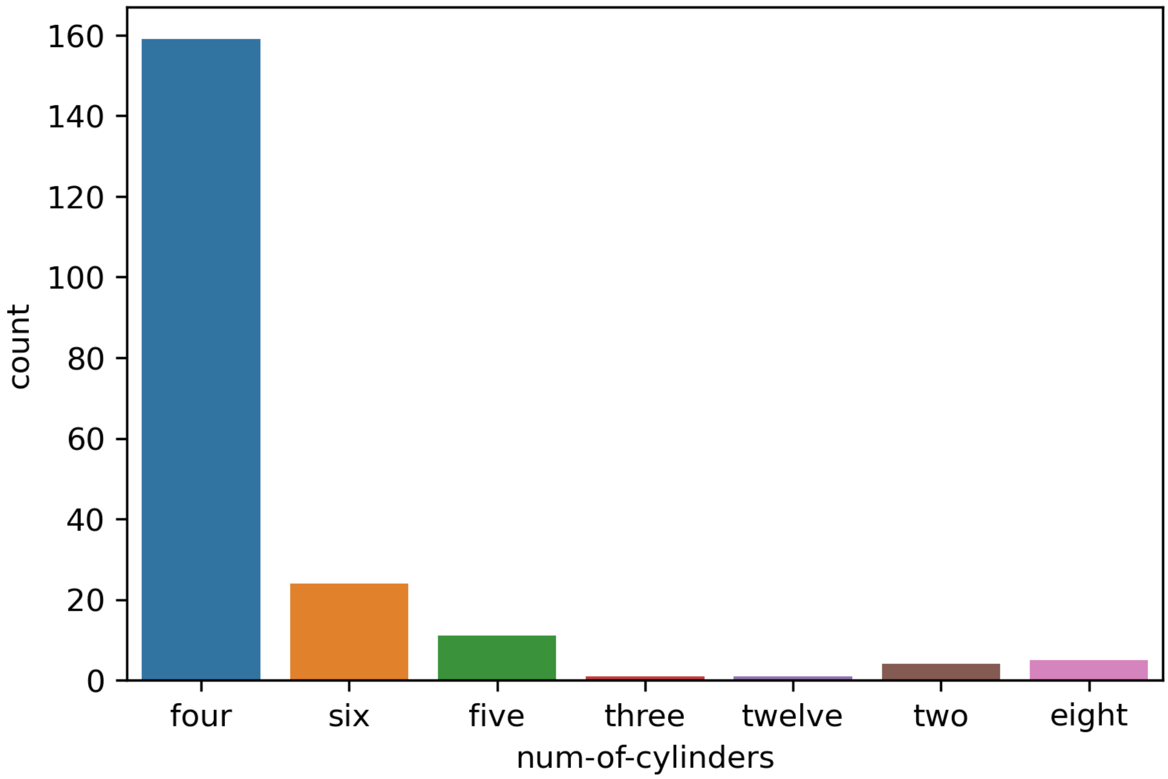
	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	117.120000	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.328484	3.254183	10.142537	104.246234
std	1.245307	33.157365	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270993	0.313721	3.972040	39.519338
min	-2.000000	65.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000
25%	0.000000	97.600000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	8.600000	70.000000
50%	1.000000	103.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000
75%	2.000000	137.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	9.400000	116.000000
max	3.000000	256.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	288.000000

peak-rpm	city-mpg	highway-mpg	price
205.000000	205.000000	205.000000	205.000000
5124.881618	25.219512	30.751220	13202.101059
477.004538	6.542142	6.886443	7868.849339
4150.000000	13.000000	16.000000	5118.000000
4800.000000	19.000000	25.000000	7788.000000
5200.000000	24.000000	30.000000	10595.000000
5500.000000	30.000000	34.000000	16500.000000
6600.000000	49.000000	54.000000	45400.000000

Following table show the min, max, average, std deviation and many more parameters.

- To determine the count of cars having different number of cylinders. A countplot graph is being plotted.

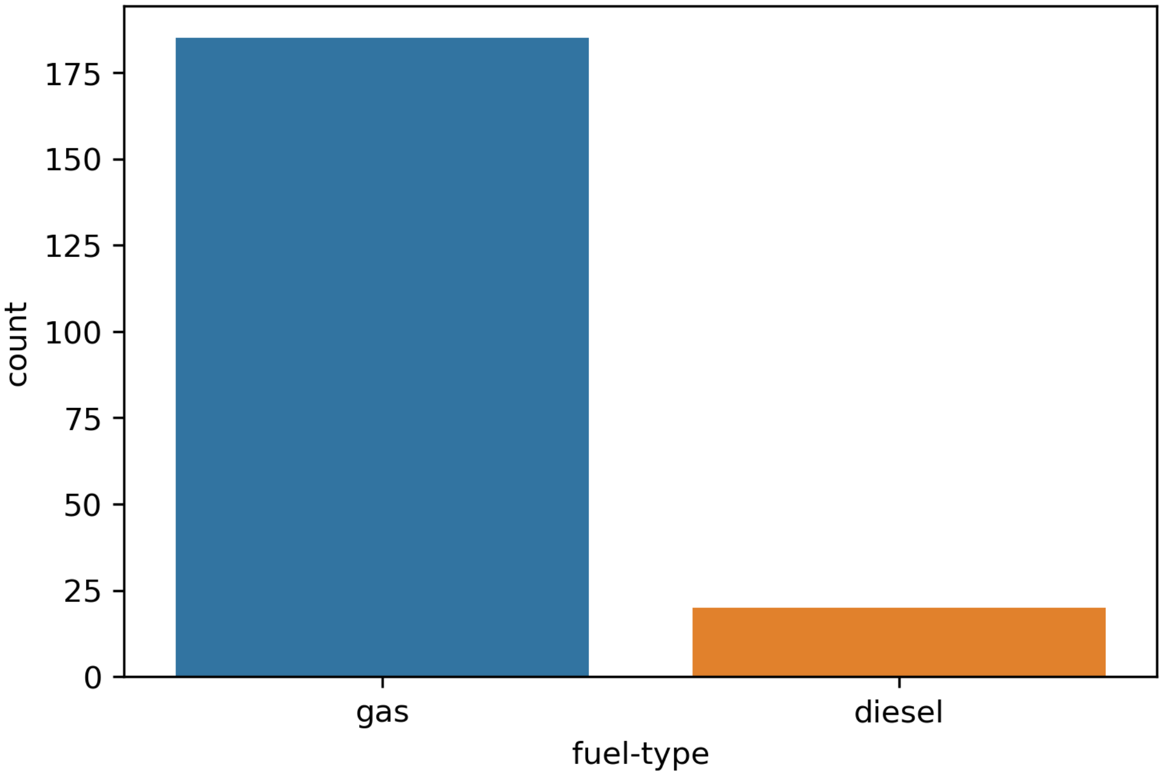
```
#determing the count of the cars having different number of cylinders
sns.countplot(x='num-of-cylinders',data=data)
```



Observed that cars having 4 cylinders is the maximum in number of counts.

- To determine the count of cars based on the type of fuel. A countplot graph is being plotted.

```
sns.countplot(x='fuel-system',data=data)
```



- Majority of Automobile almost 180 cars run on gas, and hardly 25 automobiles run on diesel.
- Which is because, automobile of diesel type can cause a lot of knocking if the automobile is ideal for a period of time, which is why customer prefer gas over diesel.

innovation.creation

Project: Determination of the price of Automobile

- Determining the correlation of parameters with respect to the price, we split the parameters into two list which help in easy comparison.
- Correlation of the parameter is determined using corr() function.

```
data[['symboling','normalized-losses','wheel-base','length','width','height','curb-weight','engine-size','price']].corr()
```

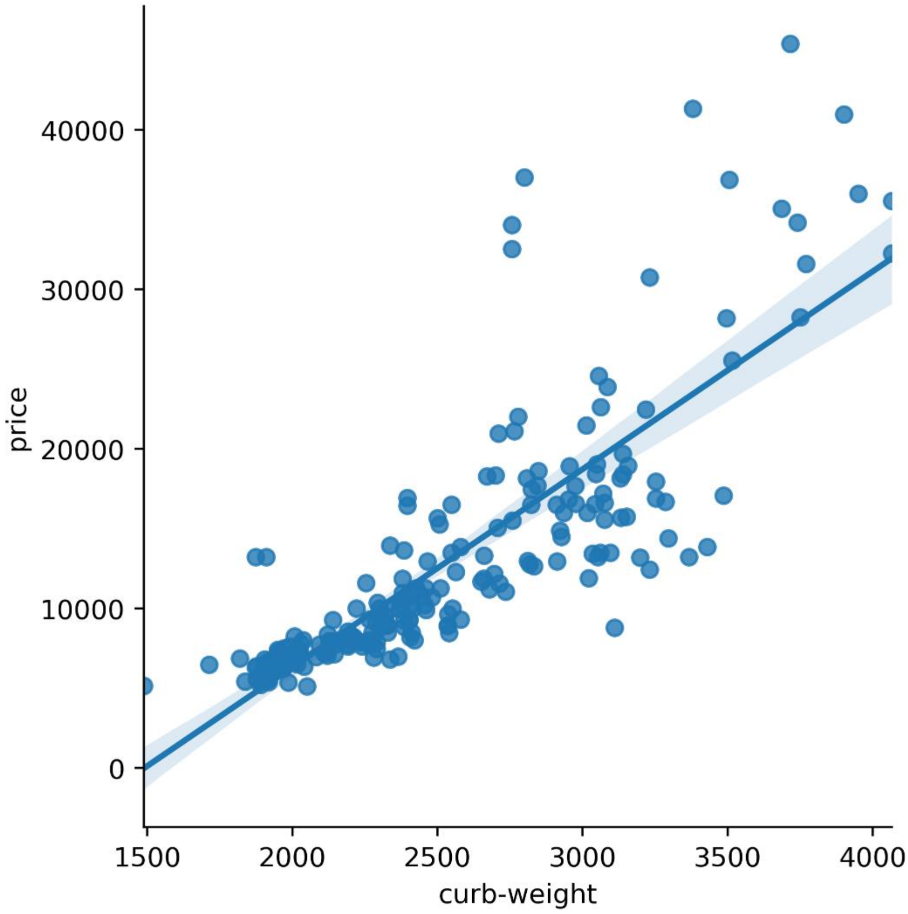
	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	price
symboling	1.000000	0.465190	-0.531954	-0.357612	-0.232919	-0.541038	-0.227691	-0.105790	-0.082201
normalized-losses	0.465190	1.000000	-0.056518	0.019209	0.084195	-0.370706	0.097785	0.110997	0.133999
wheel-base	-0.531954	-0.056518	1.000000	0.874587	0.795144	0.589435	0.776386	0.569329	0.583168
length	-0.357612	0.019209	0.874587	1.000000	0.841118	0.491029	0.877728	0.683360	0.682986
width	-0.232919	0.084195	0.795144	0.841118	1.000000	0.279210	0.867032	0.735433	0.728699
height	-0.541038	-0.370706	0.589435	0.491029	0.279210	1.000000	0.295572	0.067149	0.134388
curb-weight	-0.227691	0.097785	0.776386	0.877728	0.867032	0.295572	1.000000	0.850594	0.820825
engine-size	-0.105790	0.110997	0.569329	0.683360	0.735433	0.067149	0.850594	1.000000	0.861752
price	-0.082201	0.133999	0.583168	0.682986	0.728699	0.134388	0.820825	0.861752	1.000000

```
data[['bore','stroke','compression-ratio','horsepower','peak-rpm','city-mpg','highway-mpg','price']].corr()
```

	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
bore	1.000000	-0.055909	0.005201	0.575737	-0.254761	-0.584508	-0.586992	0.532300
stroke	-0.055909	1.000000	0.186105	0.088264	-0.066844	-0.042179	-0.043961	0.082095
compression-ratio	0.005201	0.186105	1.000000	-0.205740	-0.435936	0.324701	0.265201	0.070990
horsepower	0.575737	0.088264	-0.205740	1.000000	0.130971	-0.803162	-0.770903	0.757917
peak-rpm	-0.254761	-0.066844	-0.435936	0.130971	1.000000	-0.113723	-0.054257	-0.100854
city-mpg	-0.584508	-0.042179	0.324701	-0.803162	-0.113723	1.000000	0.971337	-0.667449
highway-mpg	-0.586992	-0.043961	0.265201	-0.770903	-0.054257	0.971337	1.000000	-0.690526
price	0.532300	0.082095	0.070990	0.757917	-0.100854	-0.667449	-0.690526	1.000000

- It is observed that curb-weight, engine-size and horsepower are strongly correlated.
- While the parameters like highway-mpg, city-mpg are inversely correlated. This can be proven by plotting wrt price.

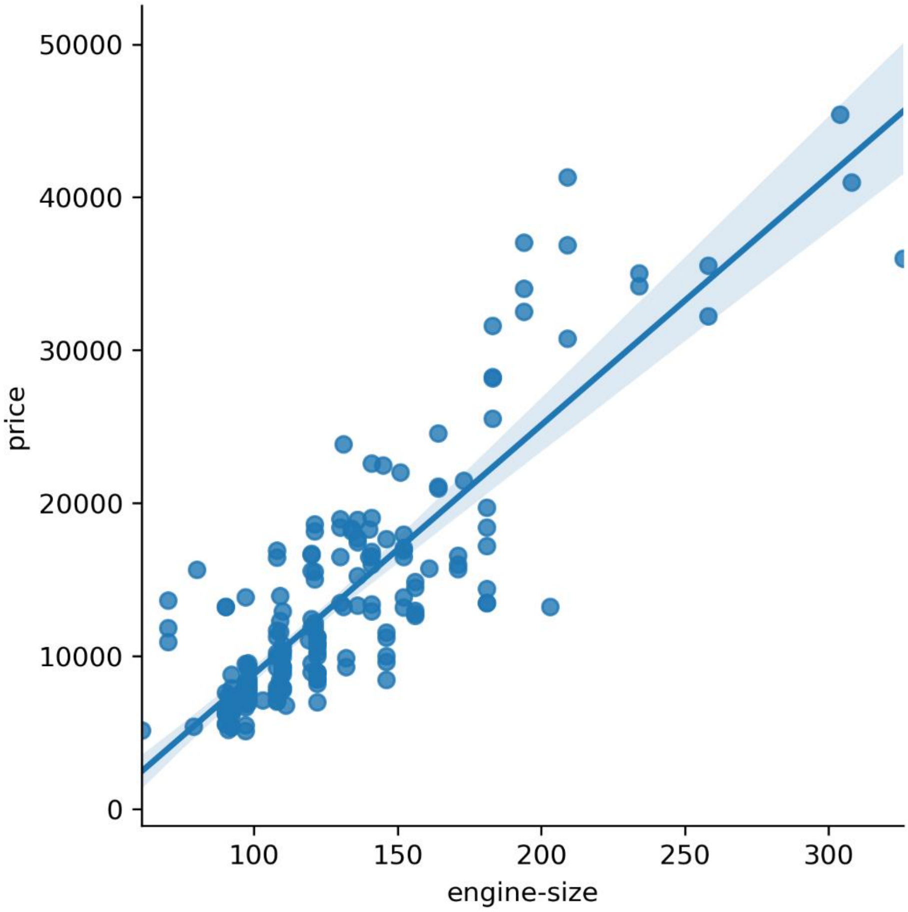
```
sns.lmplot(x='curb-weight',y='price',data=data)
plt.savefig('plot4.png', dpi=300, bbox_inches='tight')
```



With the plot we can conclude that most of the point lye on the line, also proving that curb-weight and price are strongly correlated.

- To plot the graph between engine-size vs price

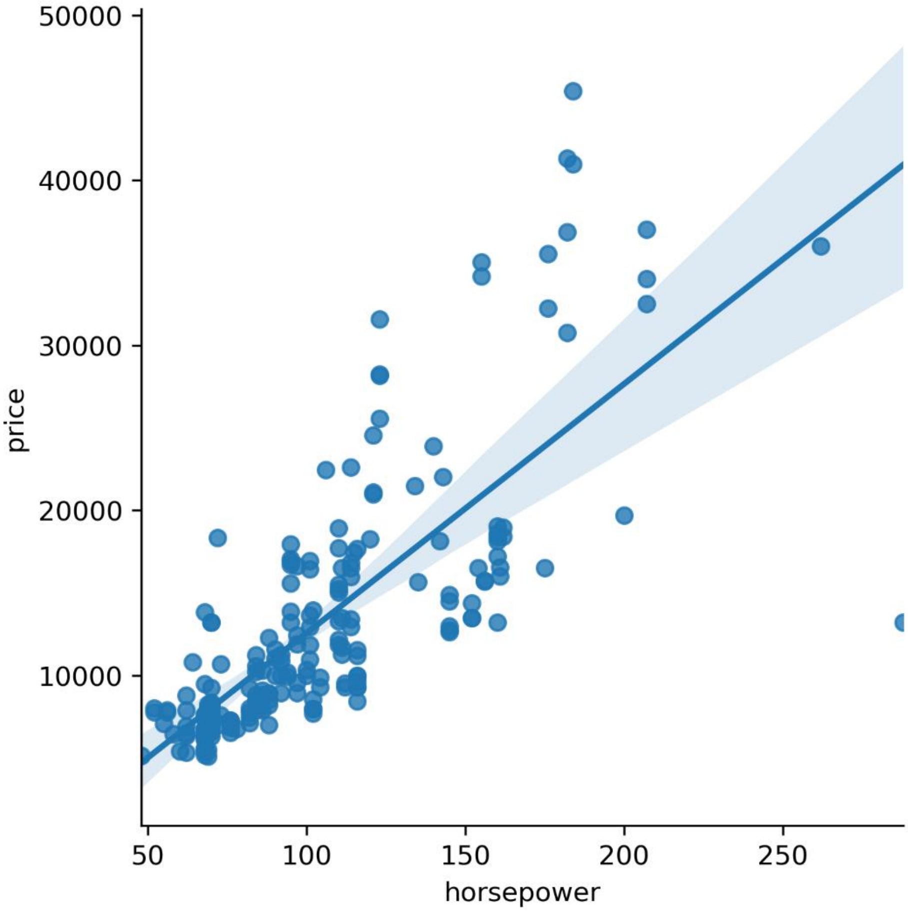
```
sns.lmplot(x='engine-size',y='price',data=data)
plt.savefig('plot5.png', dpi=300, bbox_inches='tight')
```



With the plot we can conclude that most of the point lye on the line, also proving that engine-size and price are strongly correlated.

- To plot the graph between horsepower vs price

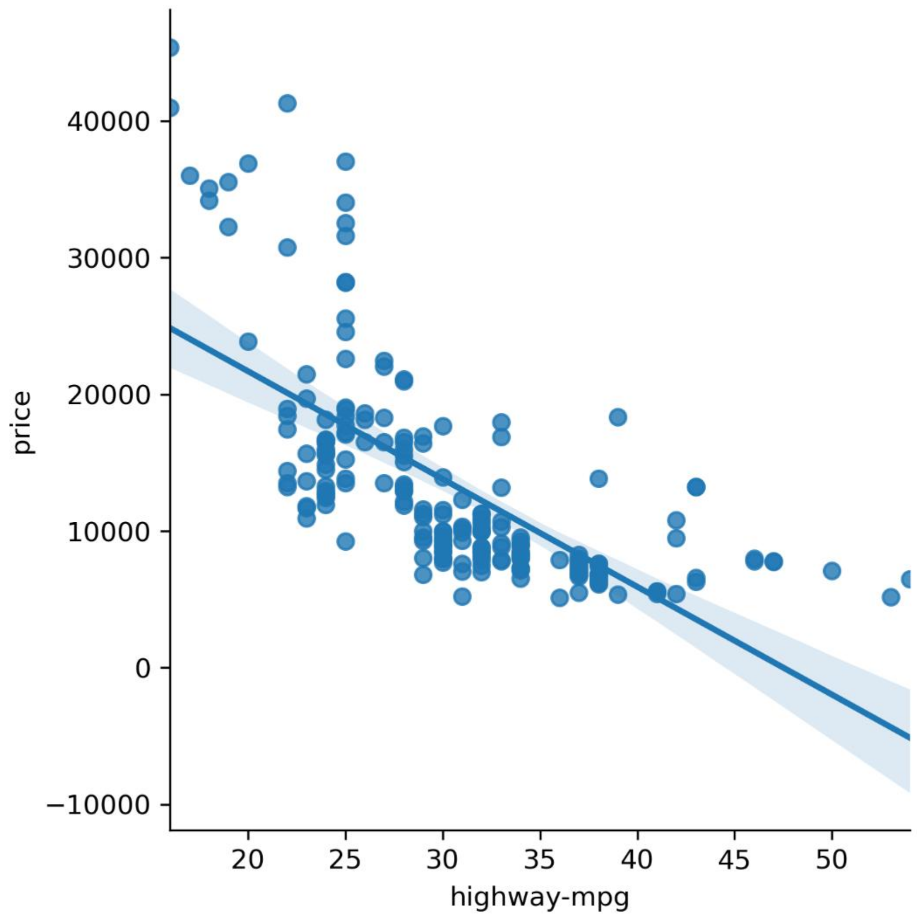
```
sns.lmplot(x='horsepower',y='price',data=data)
plt.savefig('plot6.png', dpi=300, bbox_inches='tight')
```



With the plot we can conclude that most of the point lye on the line, also proving that horsepower and price are strongly correlated.

- To plot the graph between highway-mpg vs price

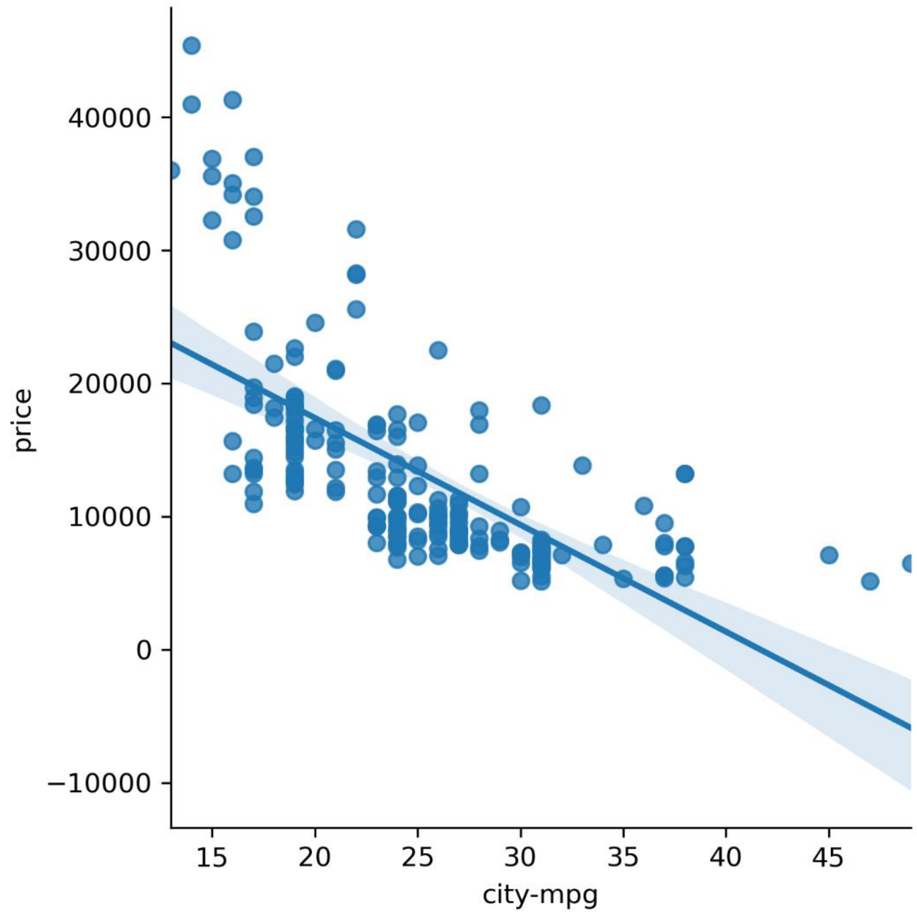
```
sns.lmplot(x='highway-mpg',y='price',data=data)
plt.savefig('plot6.png', dpi=300, bbox_inches='tight')
```



With the plot we can conclude that most of the point lye on the line, also proving that highway-mpg and price are strongly inversely correlated.

- To plot the graph between city-mpg vs price

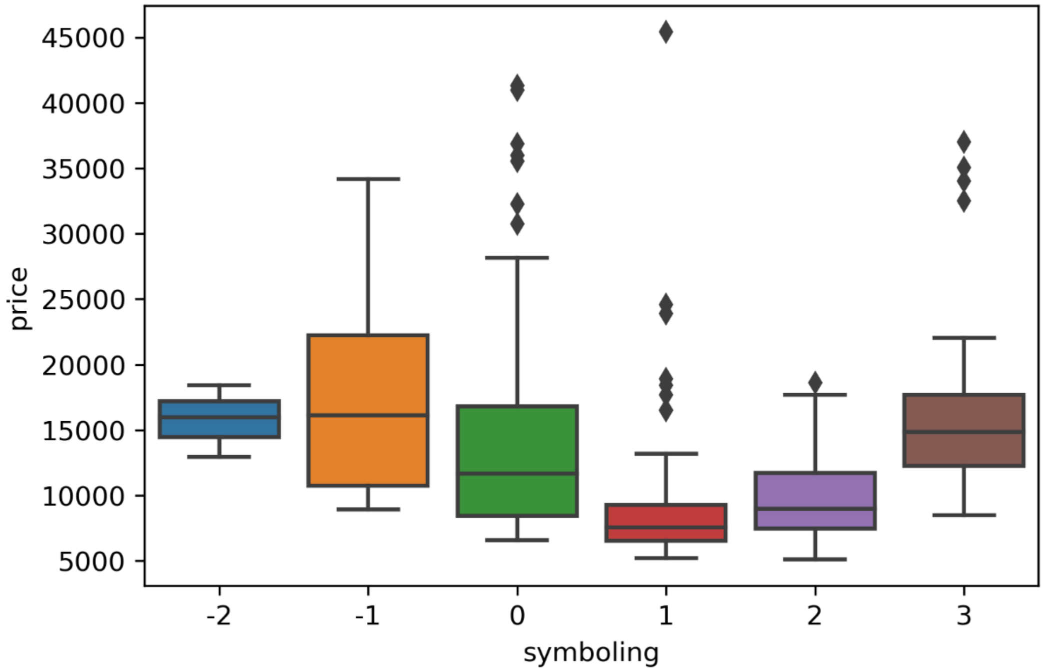
```
sns.lmplot(x='city-mpg',y='price',data=data)
plt.savefig('plot7.png', dpi=300, bbox_inches='tight')
```



- With the plot we can conclude that most of the point lye on the line, also proving that city-mpg and price are strongly inversely correlated.

- Talking about symbolling which is an important parameter in any automobile industry. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is riskier (or less), this symbol is adjusted by moving it up (or down) the scale. Actuarians call this process "symbolling". A value of +3 indicates that the auto is risky, - 3 that it is probably pretty safe.
- To check the mean, max and other parameters associated with the cars, a **Boxplot** graph is plotted between symbolling and price.

```
sns.boxplot(x='symboling',y='price',data=data)
plt.savefig('plot7.png', dpi=300, bbox_inches='tight')
```



Following shows the boxplot of the symbolling vs price. It is observed that with more safety of the car the price would increase by neglecting anomalies.

- Its is observed that they are various parameters which are in object type and has to be converted float or int type

```
#to see the updated info
data.info()
```

```
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    float64
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       203 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    float64
 19  stroke             205 non-null    float64
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    float64
 22  peak-rpm           205 non-null    float64
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    float64
dtypes: float64(11), int64(5), object(10)
memory usage: 41.8+ KB
```

- We see that they are various parameters which are of object type and has to be changed.
- Which is why **OneHotEncoder** function is being used to convert the object to int type.

```
#importing OneHotEncoder from sklearn
from sklearn.preprocessing import OneHotEncoder
```

- Parameters like symbolling, fuel-type, engine-location, aspiration, num-of-doors, body-style, make, drive-wheels, num-of-cylinders and engine-type has to be converted.
- Therefore, by using **get_dummies** function we convert those data type.

```
symbol=pd.get_dummies(data['symboling'],drop_first=True)
Fuel_type=pd.get_dummies(data['fuel-type'],drop_first=True)
engine_loc=pd.get_dummies(data['engine-location'],drop_first=True)
aspiration=pd.get_dummies(data['aspiration'],drop_first=True)
num_of_doors=pd.get_dummies(data['num-of-doors'],drop_first=True)
body_style=pd.get_dummies(data['body-style'],drop_first=True)
makes=pd.get_dummies(data['make'],drop_first=True)
drive_wheel=pd.get_dummies(data['drive-wheels'],drop_first=True)
numofcyl=pd.get_dummies(data['num-of-cylinders'],drop_first=True)
engine_type=pd.get_dummies(data['engine-type'],drop_first=True)
```

Note: We are doing this because we need these values to be distinct, so we get columns of these distinct parameters, which helps in training and testing data.

- Now these existing columns has to be dropped(removed) from the data sets, and the columns obtained from one hot encoder has to be concatenated.

```
data.drop(['fuel-type','engine-location','aspiration','num-of-doors','body-style','make','drive-wheels','num-of-cylinders','engine-type'],axis=1,inplace=True)
```

- I have select the parameters of object type and now they are being dropped.

```
data=pd.concat([data,Fuel_type,engine_loc,aspiration,num_of_doors,body_style,makes,drive_wheel,numofcyl,engine_type],axis=1)
```

- So we concatenate the new columns into the dataset using **pd.concat** function.

Preparing Machine Learning Model

- In order to train or test these data sets we need to segregate these data, in order to do so we use the function **train_test_split.**
- This function actually helps us to segregate the data based on training and testing datasets used for processing through algorithm.

```
#import train_test split library from sklearn

from sklearn.model_selection import train_test_split
```

- Before giving the datasets to train and test function, we need to figure out what will be the ‘x’(inputs) and ‘y’(output).
- Let all the columns except ‘fuel system’ and ‘price’ be the input(x) and since the price is to be predicted, output(y) is price.

```
X_train, X_test, y_train, y_test=train_test_split(data.drop(['fuel-system','price'],axis=1),data['price'],test_size=0.2)
```

- Here we give 80%of data to train and the remaining 20% is to be tested, to check the accuracy of the algorithm. More the data is being trained, more the precision and accuracy.
- Since the predicted value is continuous it comes under linear regression method, therefore it is necessary to choose algorithm of regression.
- Various algorithm like Linear regression, Decision Trees, Random Forest regressor, AdaBoost Regressor, Gradient boosting Regressor, etc.

Linear Regression

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression.

- Import **linear regression** from sklearn(Scikit)

```
#import linear regression from sklearn.linear model

from sklearn.linear_model import LinearRegression

#intialise linear regression as lm

lm=LinearRegression()
```

- Now the dataset which has to be trained will be fit using **.fit** method and predict the value with the help of testing datasets using **.predict** method

```
lm.fit(X_train,y_train)
pred=lm.predict(X_test)
```

- Now it's the time to determine the accuracy of the given algorithm using metrics, which is being imported from sklearn.
- In metrics we use r2_score to determine the accuracy of the given algorithm.

```
#to know the accuracy
from sklearn import metrics
print('R2_score:',metrics.r2_score(y_test,pred))
```

```
R2_score: 0.7935854182893958
```

Decision Trees

A decision tree is a supervised machine learning model used to predict a target by learning decision rules from features. As the name suggests, we can think of this model as breaking down our data by making a decision based on asking a series of questions.

- Import **decision tree** from sklearn(Scikit)

```
#import decision tree from sklearn
from sklearn import tree
#initialize decisionTreeRegressor as reg
reg=DecisionTreeRegressor()
```

- Now the dataset which has to be trained will be fit using **.fit** method and predict the value with the help of testing datasets using **.predict** method

```
dt=dt.fit(X_train,y_train)
pred=dt.predict(X_test)
```

- Now it's the time to determine the accuracy of the given algorithm using metrics, which is being imported from sklearn.
- In metrics we use r2_score to determine the accuracy of the given algorithm.

```
#to know the accuracy
from sklearn import metrics
print('R2_score:',metrics.r2_score(y_test,pred))
```

```
R2_score: 0.9903312780153448
```

RandomForest Regressor

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging.

- Import **RandomForest Regressor** from sklearn(Scikit)

```
#import RandomForest Regressor from sklearn
from sklearn.ensemble import RandomForestRegressor
#initialize RandomForestRegressor as reg
reg=reg.DecisionTreeRegressor()
```

- Now the dataset which has to be trained will be fit using **.fit** method and predict the value with the help of testing datasets using **.predict** method

```
reg.fit(X_train,y_train)
pred=reg.predict(X_test)
```

- Now it's the time to determine the accuracy of the given algorithm using metrics, which is being imported from sklearn.
- In metrics we use r2_score to determine the accuracy of the given algorithm.

```
#to know the accuracy
from sklearn import metrics
print('R2_score:',metrics.r2_score(y_test,pred))
```

```
R2_score: 0.9417571920145558
```

AdaBoost Regressor

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

- Import **AdaBoostRegressor** Regressor from sklearn(Scikit)

```
#import AdaBoostRegressor from sklearn.ensemble model

from sklearn.ensemble import AdaBoostRegressor

#intialise AdaBoostRegressor as abr

abr= AdaBoostRegressor()
```

- Now the dataset which has to be trained will be fit using **.fit** method and predict the value with the help of testing datasets using **.predict** method

```
reg.fit(X_train,y_train)
pred=reg.predict(X_test)
```

- Now it's the time to determine the accuracy of the given algorithm using metrics, which is being imported from sklearn.
- In metrics we use r2_score to determine the accuracy of the given algorithm.

```
#to know the accuracy
from sklearn import metrics
print('R2_score:',metrics.r2_score(y_test,pred))
```

```
R2_score: 0.9247065952548518
```

GradientBoostingRegressor

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

- Import **GradientBoosting Regressor** from sklearn(Scikit)

```
#import GradientBoostingRegressor from sklearn.ensemble model

from sklearn.ensemble import GradientBoostingRegressor

#intialise GradientBoostingRegressor as GBR

GBR= GradientBoostingRegressor()
```

- Now the dataset which has to be trained will be fit using **.fit** method and predict the value with the help of testing datasets using **.predict** method

```
GBR.fit(X_train,y_train)
pred=GBR.predict(X_test)
```

- Now it's the time to determine the accuracy of the given algorithm using metrics, which is being imported from sklearn.
- In metrics we use r2_score to determine the accuracy of the given algorithm.

```
#to know the accuracy
from sklearn import metrics
print('R2_score:',metrics.r2_score(y_test,pred))
```

```
R2_score: 0.9400643433571121
```

ML Model Chart

The following table will give clarity on which ML algorithm to be used to get higher accuracy in predicting the price of the automobile.

We use r2_score since the given problem deals with linear regression which can take only the r2_score to predict values.

I have considered five ML algorithms which are as follows

- Decision Tree
- RandomForest Regressor
- GradientBoostingRegressor
- AdaBoostRegressor
- Linear regression

Serial number	**ML algorithm name**	**metric used to evaluate the model**	**metric score**
1	Decision Tree	r2_score	0.9903312780153448
2	RandomForest Regressor	r2_score	0.9417571920145558
3	GradientBoostingRegressor	r2_score	0.9400643433571121
4	AdaBoostRegressor	r2_score	0.9247065952548518
5	Linear regression	r2_score	0.8577571733103396

Hurdles

In the Explanatory Data Analysis, I found it difficult to convert those unknown values, and then I used NumPy to remove all the unknowns using np.Nan, because I initially converted those unknowns('?') to 0 but there was a variation in mean and the variance. This is why I converted those unknowns to np.Nan,

The mean of columns has to be determined and is being replaced to the unknown, but it is not possible to replace the unknown of certain parameters like num-of-doors, which is why I had to find the mode of those values through bar-graph.

I found hurdles looking for the syntax of different regression methods since the majority of them belonged to the classifier class. It was time-consuming for me to convert all the object types to int/float which involves a sequence of steps that's to convert into distinct columns, dropping the existing columns of the table and concatenate the new columns.

It took some effort for me to convert those object types into float, further, any calculation can be done so to replace those values.

They were few challenges for me, firstly to look for different regression method with high accuracy that is r2_score must be nearly 1, which was challenging because I had to explore regression algorithms which I wasn't aware of and had to look for the syntax of it and also see its accuracy/r2_score to check if it's suitable for the following case.

Conclusion

From the given dataset we have to predict the price of the automobile as discussed earlier, to determine the price, there are various factors involved in it. Firstly, we need to use the linear regression method since the price of the automobile is continuous, which means that different linear regression methods like Decision Tree, RandomForest Regressor, GradientBoostingRegressor, AdaBoostRegressor, Linear regression, etc has to be carried and verify which algorithm is appropriate for the given case.

From the following table, a model chart of the different algorithm is used through their r2_score and it is observed that the Decision tree is the best algorithm that can be used to predict the price of the automobile because of its r2_score which is nearly 1, which means that test value and the predicted value are strongly correlated.

Followed by randomForest Regressor and gradient boosting regressor sharing the same r2_score of 0.94.

It is not ideal to choose to linear regression algorithm because of its poor r2_score, therefore it is very difficult to predict the price of the automobile as it is not accurate.

Biblography

1. https://scikit-learn.org/stable
2. https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_regression.html
3. https://www.geeksforgeeks.org/random-forest-regression-in-python/
4. https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054
5. https://numpy.org/doc/stable/user/misc.html
6. https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
7. https://scikit-learn.org/stable/modules/tree.html#regression