```
import numpy as np
import pandas as pd
#Visualization
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score


# 1. Problem Statement
#To predict price of cars using various features


# 2. Data Gathering
df = pd.read_csv("/content/autos_dataset.csv")
df
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine
0	3	?	alfa-romero	gas	std	two	convertible	rwd	fro
1	3	?	alfa-romero	gas	std	two	convertible	rwd	fro
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	fro
3	2	164	audi	gas	std	four	sedan	fwd	fro
4	2	164	audi	gas	std	four	sedan	4wd	fro
...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	gas	std	four	sedan	rwd	fro
201	-1	95	volvo	gas	turbo	four	sedan	rwd	fro
202	-1	95	volvo	gas	std	four	sedan	rwd	fro
203	-1	95	volvo	diesel	turbo	four	sedan	rwd	fro
204	-1	95	volvo	gas	turbo	four	sedan	rwd	fro

205 rows × 26 columns

```
df.head() # head() function is generally use for to print first 5 rows bydefault
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136

5 rows × 26 columns

```
df.head().T # after use of T after head() function, Columns goes to rows place and Rows goes to Columns place
```

	0	1	2	3	4
symboling	3	3	1	2	2
normalized-losses	?	?	?	164	164
make	alfa-romero	alfa-romero	alfa-romero	audi	audi
fuel-type	gas	gas	gas	gas	gas
aspiration	std	std	std	std	std
num-of-doors	two	two	two	four	four
body-style	convertible	convertible	hatchback	sedan	sedan
drive-wheels	rwd	rwd	rwd	fwd	4wd
engine-location	front	front	front	front	front
wheel-base	88.6	88.6	94.5	99.8	99.4
length	168.8	168.8	171.2	176.6	176.6
width	64.1	64.1	65.5	66.2	66.4
height	48.8	48.8	52.4	54.3	54.3
curb-weight	2548	2548	2823	2337	2824
engine-type	dohc	dohc	ohcv	ohc	ohc
num-of-cylinders	four	four	six	four	five
engine-size	130	130	152	109	136
fuel-system	mpfi	mpfi	mpfi	mpfi	mpfi
bore	3.47	3.47	2.68	3.19	3.19
stroke	2.68	2.68	3.47	3.4	3.4
compression-ratio	9.0	9.0	9.0	10.0	8.0
horsepower	111	111	154	102	115

```
# 3. Exploratory Data Analysis (EDA) :
```

```
df.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
```

```
df.isna().sum() # Checking there is avl. of any NaN values
```

```
symboling           0
normalized-losses    0
make                0
fuel-type           0
aspiration          0
num-of-doors        0
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                0
stroke              0
compression-ratio    0
horsepower          0
peak-rpm            0
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

```
"""Check all the columns one by one, fill the NaN values , Convert "object" Data Types into "int" or "float" bcz during
Model Training and Model valuation Calculation , it is must requirement of integers"""
```

```
'Check all the columns one by one, fill the NaN values , Convert "object" Data Types in
to "int" or "float" bcz during\nModel Training and Model valuation Calculation , it is
must requirement of integers'
```

```
# 3.1 Symboling
```

```
df["symboling"]
```

```
0      3
1      3
2      1
3      2
4      2
      ..
200   -1
201   -1
202   -1
203   -1
204   -1
Name: symboling, Length: 205, dtype: int64
```

```
# 3.2 normalized-losses
```

```
df["normalized-losses"]
```

```
0        ?
1        ?
2        ?
3      164
4      164
      ...
200     95
201     95
202     95
203     95
204     95
Name: normalized-losses, Length: 205, dtype: object
```

```
"""In the return output we see question marks and its data type is "object" , so this ? qustion mark indicate it is NaN Value.
It is necessary to convert this ? question marks to NaN standard format.
Because it is generate some confusion because its data type is "object" and at isna().sum() function there is also shows
there is not available of NaN values.
So we replace ? Question mark into NaN Value."""
```

```
'In the return output we see question marks and its data type is "object" , so this ? q
ustion mark indicate it is NaN Value.\nIt is necessary to convert this ? question marks
to NaN standard format.\nBecause it is generate some confusion because its data type is
"object" and at isna().sum() function there is also shows\nthere is not available of Na
```

```
df.replace({"?":np.nan},inplace=True) # By using replace function we replace all question mark with NaN Values
```

```
df.isna().sum() # After replace of ? question mark we find summary of availabiliy of NaN Values.
```

```
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
dtype: int64
```

```
df.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  164 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       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               201 non-null    object 
 19  stroke             201 non-null    object 
 20  compression-ratio  205 non-null    float64
 21  horsepower         203 non-null    object 
 22  peak-rpm           203 non-null    object 
 23  city-mpg           205 non-null    int64  
 24  highway-mpg        205 non-null    int64  
 25  price              201 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

```
df["normalized-losses"] = df["normalized-losses"].astype(float) # "normalized-losses" this column in "object" data type
 # So we converted into "float" data type


df["normalized-losses"].mean()
```

```
122.0
```

```
df["normalized-losses"].median()
```

```
115.0
```

```
"""We check both Mean and Median for "normalized-losses" column but Mean value is slighter greater than Median.
This is because of availablity of some Outliers.
So we use Median to fill the NaN Values."""
```

```
'We check both Mean and Median for "normalized-losses" column but Mean value is slighte
r greater than Median.\nThis is because of availablity of some Outliers.\nSo we use Med
ian to fill the NaN Values.'
```

```
df["normalized-losses"] = df["normalized-losses"].fillna(df["normalized-losses"].median()).astype(int)
# we fill the NaN Values by Median of "normalized-losses" column and converted into "int" data type


df["normalized-losses"]
```

```
0      115
1      115
2      115
3      164
4      164
      ...
200     95
201     95
202     95
203     95
204     95
Name: normalized-losses, Length: 205, dtype: int64
```

```
df.info() # By doing Type Casting we convert "object" into "int" data type and
 # We fill the NaN Values by Median of "normalized-losses" column
# Like wise that we process all Columns( Fill NaN Values and Converted into "int" or "float")
```

```
<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    int64
 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               201 non-null    object
 19  stroke             201 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         203 non-null    object
 22  peak-rpm           203 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              201 non-null    object
dtypes: float64(5), int64(6), object(15)
memory usage: 41.8+ KB
```

```
# 3.3 make
df["make"]
```

```
0      alfa-romero
1      alfa-romero
2      alfa-romero
3             audi
4             audi
          ...
200          volvo
201          volvo
202          volvo
203          volvo
204          volvo
Name: make, Length: 205, dtype: object
```

```
df["make"].nunique() # We find out how many type of unique values availale in " make " column
```

```
22
```

```
df["make"].value_counts() # We find out how many type of unique values availale and its Count
```

```
toyota           32
nissan           18
mazda            17
mitsubishi       13
honda            13
volkswagen       12
subaru           12
peugot           11
volvo            11
dodge             9
mercedes-benz     8
bmw               8
audi              7
plymouth          7
saab              6
porsche           5
isuzu             4
jaguar            3
chevrolet         3
alfa-romero       3
renault           2
mercury           1
Name: make, dtype: int64
```

```
#3.4 fuel-type
```

```
df["fuel-type"].value_counts()
```

```
gas       185
diesel     20
Name: fuel-type, dtype: int64
```

```
df["fuel-type"].replace({'gas' : 0 , 'diesel' : 1}, inplace = True)
```

```
df["fuel-type"].value_counts()
```

```
0    185
1     20
Name: fuel-type, dtype: int64
```

```
df.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    int64
 2   make               205 non-null    object
 3   fuel-type          205 non-null    int64
 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               201 non-null    object
 19  stroke             201 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         203 non-null    object
 22  peak-rpm           203 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              201 non-null    object
dtypes: float64(5), int64(7), object(14)
memory usage: 41.8+ KB
```

```
# 3.5 aspiration
```

```
df['aspiration'].value_counts()
```

```
std      168
turbo     37
Name: aspiration, dtype: int64
```

```
df['aspiration'].replace({'std' : 0 , 'turbo' : 1}, inplace = True)
```

```
df['aspiration'].value_counts()
```

```
0    168
1     37
Name: aspiration, dtype: int64
```

```
# 3.6 num-of-doors
```

```
df['num-of-doors'].value_counts()
```

```
four    114
two      89
Name: num-of-doors, dtype: int64
```

```
df['num-of-doors'].replace({'four' : 4 , 'two' : 2}, inplace = True)
df['num-of-doors'].value_counts()
```

```
4.0    114
2.0     89
Name: num-of-doors, dtype: int64
```

```
df['num-of-doors'].mean()
```

```
3.123152709359606
```

```
df['num-of-doors'].median()
```

```
4.0
```

```
df['num-of-doors'].mode()[0]
```

```
4.0
```

```
df['num-of-doors'].fillna(df['num-of-doors'].median() , inplace = True)
```

```
##3.7 body-style
```

```
df['body-style'].value_counts()
```

```
sedan          96
hatchback      70
wagon          25
hardtop         8
convertible     6
Name: body-style, dtype: int64
```

```
df.T
```

	0	1	2	3	4	5	6	7	8	9	...	195	196	197	198
symboling	3	3	1	2	2	2	1	1	1	0	...	-1	-2	-1	-2
normalized-losses	115	115	115	164	164	115	158	115	158	115	...	74	103	74	103
make	alfa-romero	alfa-romero	alfa-romero	audi	audi	audi	audi	audi	audi	audi	...	volvo	volvo	volvo	volvo
fuel-type	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
aspiration	0	0	0	0	0	0	0	0	1	1	...	0	0	0	1
num-of-doors	2.0	2.0	2.0	4.0	4.0	2.0	4.0	4.0	4.0	2.0	...	4.0	4.0	4.0	4.0
body-style	convertible	convertible	hatchback	sedan	sedan	sedan	sedan	wagon	sedan	hatchback	...	wagon	sedan	wagon	sedan
drive-wheels	rwd	rwd	rwd	fwd	4wd	fwd	fwd	fwd	fwd	4wd	...	rwd	rwd	rwd	rwd
engine-location	front	front	front	front	front	front	front	front	front	front	...	front	front	front	front
wheel-base	88.6	88.6	94.5	99.8	99.4	99.8	105.8	105.8	105.8	99.5	...	104.3	104.3	104.3	104.3
length	168.8	168.8	171.2	176.6	176.6	177.3	192.7	192.7	192.7	178.2	...	188.8	188.8	188.8	188.8
width	64.1	64.1	65.5	66.2	66.4	66.3	71.4	71.4	71.4	67.9	...	67.2	67.2	67.2	67.2
height	48.8	48.8	52.4	54.3	54.3	53.1	55.7	55.7	55.9	52.0	...	57.5	56.2	57.5	56.2
curb-weight	2548	2548	2823	2337	2824	2507	2844	2954	3086	3053	...	3034	2935	3042	3045
engine-type	dohc	dohc	ohcv	ohc	ohc	ohc	ohc	ohc	ohc	ohc	...	ohc	ohc	ohc	ohc
num-of-cylinders	four	four	six	four	five	five	five	five	five	five	...	four	four	four	four
engine-size	130	130	152	109	136	136	136	136	131	131	...	141	141	141	130
fuel-system	mpfi	mpfi	mpfi	mpfi	mpfi	mpfi	mpfi	mpfi	mpfi	mpfi	...	mpfi	mpfi	mpfi	mpfi
bore	3.47	3.47	2.68	3.19	3.19	3.19	3.19	3.19	3.13	3.13	...	3.78	3.78	3.78	3.62
stroke	2.68	2.68	3.47	3.4	3.4	3.4	3.4	3.4	3.4	3.4	...	3.15	3.15	3.15	3.15
compression-ratio	9.0	9.0	9.0	10.0	8.0	8.5	8.5	8.5	8.3	7.0	...	9.5	9.5	9.5	7.5
horsepower	111	111	154	102	115	110	110	110	140	160	...	114	114	114	162
peak-rpm	5000	5000	5000	5500	5500	5500	5500	5500	5500	5500	...	5400	5400	5400	5100
city-mpg	21	21	19	24	18	19	19	19	17	16	...	23	24	24	17
highway-mpg	27	27	26	30	22	25	25	25	20	22	...	28	28	28	22
price	13495	16500	16500	13950	17450	15250	17710	18920	23875	NaN	...	13415	15985	16515	18420

26 rows × 205 columns

```
df = pd.get_dummies(df,columns = ['body-style'])
df
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	drive-wheels	engine-location	wheel-base	length	...	horsepower	peak-rpm	city-mpg	highway-mpg
0	3	115	alfa-romero	0	0	2.0	rwd	front	88.6	168.8	...	111	5000	21	27
1	3	115	alfa-romero	0	0	2.0	rwd	front	88.6	168.8	...	111	5000	21	27
2	1	115	alfa-romero	0	0	2.0	rwd	front	94.5	171.2	...	154	5000	19	26
3	2	164	audi	0	0	4.0	fwd	front	99.8	176.6	...	102	5500	24	30
4	2	164	audi	0	0	4.0	4wd	front	99.4	176.6	...	115	5500	18	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
201	-1	95	volvo	0	1	4.0	rwd	front	109.1	188.8	...	160	5300	19	25

```
#3.8 drive-wheels
df['drive-wheels'].value_counts()
```

```
fwd    120
rwd     76
4wd      9
Name: drive-wheels, dtype: int64
```

```
df['drive-wheels'].replace({"fwd" : 0 , "rwd" : 1 , "4wd" : 2}, inplace=True)
df['drive-wheels'].value_counts()
```

```
0    120
1     76
2      9
Name: drive-wheels, dtype: int64
```

```
# 3.9 engine-location#

df['engine-location'].value_counts()
```

```
front    202
rear       3
Name: engine-location, dtype: int64
```

```
df['engine-location'].replace({"front" : 0 , "rear" : 1 }, inplace=True)
df['engine-location'].value_counts()
```

```
0    202
1      3
Name: engine-location, dtype: int64
```

```
# 3.14 engine-type
df['engine-type'].value_counts()
```

```
ohc      148
ohcf      15
ohcv      13
dohc      12
l         12
rotor      4
dohcv      1
Name: engine-type, dtype: int64
```

```
df = pd.get_dummies(df, columns=['engine-type'])
df
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	drive-wheels	engine-location	wheel-base	length	...	body-style_hatchback	body-style_sedan	styl
0	3	115	alfa-romero	0	0	2.0	1	0	88.6	168.8	...	0	0	
1	3	115	alfa-romero	0	0	2.0	1	0	88.6	168.8	...	0	0	
2	1	115	alfa-romero	0	0	2.0	1	0	94.5	171.2	...	1	0	
3	2	164	audi	0	0	4.0	0	0	99.8	176.6	...	0	1	
4	2	164	audi	0	0	4.0	2	0	99.4	176.6	...	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	0	0	4.0	1	0	109.1	188.8		0	1	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 36 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   symboling               205 non-null    int64  
 1   normalized-losses       205 non-null    int64  
 2   make                    205 non-null    object 
 3   fuel-type               205 non-null    int64  
 4   aspiration              205 non-null    int64  
 5   num-of-doors            205 non-null    float64
 6   drive-wheels            205 non-null    int64  
 7   engine-location         205 non-null    int64  
 8   wheel-base              205 non-null    float64
 9   length                  205 non-null    float64
 10  width                   205 non-null    float64
 11  height                  205 non-null    float64
 12  curb-weight             205 non-null    int64  
 13  num-of-cylinders        205 non-null    object 
 14  engine-size             205 non-null    int64  
 15  fuel-system             205 non-null    object 
 16  bore                    201 non-null    object 
 17  stroke                  201 non-null    object 
 18  compression-ratio       205 non-null    float64
 19  horsepower              203 non-null    object 
 20  peak-rpm                203 non-null    object 
 21  city-mpg                205 non-null    int64  
 22  highway-mpg             205 non-null    int64  
 23  price                   201 non-null    object 
 24  body-style_convertible  205 non-null    uint8  
 25  body-style_hardtop      205 non-null    uint8  
 26  body-style_hatchback    205 non-null    uint8  
 27  body-style_sedan        205 non-null    uint8  
 28  body-style_wagon        205 non-null    uint8  
 29  engine-type_dohc        205 non-null    uint8  
 30  engine-type_dohcv       205 non-null    uint8  
 31  engine-type_l           205 non-null    uint8  
 32  engine-type_ohc         205 non-null    uint8  
 33  engine-type_ohcf        205 non-null    uint8  
 34  engine-type_ohcv        205 non-null    uint8  
 35  engine-type_rotor       205 non-null    uint8  
dtypes: float64(6), int64(10), object(8), uint8(12)
memory usage: 41.0+ KB
```

```
# 3.15 num-of-cylinders
```

```
df['num-of-cylinders'].value_counts()
```

```
four      159
six        24
five       11
eight       5
two         4
three       1
twelve      1
Name: num-of-cylinders, dtype: int64
```

```
df['num-of-cylinders'].replace({'four' : 4 , 'six' : 6 , 'five' : 5 , 'eight' : 8 ,
 'two' : 2, 'three' : 3,'twelve' : 12}, inplace = True)
```

```
df['num-of-cylinders'].value_counts()
```

```
4     159
6      24
5      11
8       5
2       4
3       1
12      1
Name: num-of-cylinders, dtype: int64
```

```
df
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	drive-wheels	engine-location	wheel-base	length	...	body-style_hatchback	body-style_sedan	styl
0	3	115	alfa-romero	0	0	2.0	1	0	88.6	168.8	...	0	0	
1	3	115	alfa-romero	0	0	2.0	1	0	88.6	168.8	...	0	0	
2	1	115	alfa-romero	0	0	2.0	1	0	94.5	171.2	...	1	0	
3	2	164	audi	0	0	4.0	0	0	99.8	176.6	...	0	1	
4	2	164	audi	0	0	4.0	2	0	99.4	176.6	...	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	0	0	4.0	1	0	109.1	188.8	...	0	1	
201	-1	95	volvo	0	1	4.0	1	0	109.1	188.8	...	0	1	
202	-1	95	volvo	0	0	4.0	1	0	109.1	188.8	...	0	1	
203	-1	95	volvo	1	1	4.0	1	0	109.1	188.8	...	0	1	
204	-1	95	volvo	0	1	4.0	1	0	109.1	188.8	...	0	1	

205 rows × 36 columns

```
# 3.17 fuel-system
```

```
df = pd.get_dummies(df, columns = ['fuel-system'] )
df
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	drive-wheels	engine-location	wheel-base	length	...	engine-type_ohcv	engine-type_rotor	fuel-system_1bbl	
0	3	115	alfa-romero	0	0	2.0	1	0	88.6	168.8	...	0	0	0	
1	3	115	alfa-romero	0	0	2.0	1	0	88.6	168.8	...	0	0	0	
2	1	115	alfa-romero	0	0	2.0	1	0	94.5	171.2	...	1	0	0	
3	2	164	audi	0	0	4.0	0	0	99.8	176.6	...	0	0	0	
4	2	164	audi	0	0	4.0	2	0	99.4	176.6	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95	volvo	0	0	4.0	1	0	109.1	188.8	...	0	0	0	
201	-1	95	volvo	0	1	4.0	1	0	109.1	188.8	...	0	0	0	
202	-1	95	volvo	0	0	4.0	1	0	109.1	188.8	...	1	0	0	
203	-1	95	volvo	1	1	4.0	1	0	109.1	188.8	...	0	0	0	
204	-1	95	volvo	0	1	4.0	1	0	109.1	188.8	...	0	0	0	

205 rows × 43 columns

```
df.T
```

	0	1	2	3	4	5	6	7	8	9	...	195	196	197	198
symboling	3	3	1	2	2	2	1	1	1	0	...	-1	-2	-1	-2
normalized-losses	115	115	115	164	164	115	158	115	158	115	...	74	103	74	103
make	alfa-romero	alfa-romero	alfa-romero	audi	audi	audi	audi	audi	audi	audi	...	volvo	volvo	volvo	volvo
fuel-type	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0
aspiration	0	0	0	0	0	0	0	0	1	1	...	0	0	0	1
num-of-doors	2.0	2.0	2.0	4.0	4.0	2.0	4.0	4.0	4.0	2.0	...	4.0	4.0	4.0	4.0
drive-wheels	1	1	1	0	2	0	0	0	0	2	...	1	1	1	1

```
# 3.18 bore
```

wheel-base	88.6	88.6	94.5	99.8	99.4	99.8	105.8	105.8	105.8	99.5	...	104.3	104.3	104.3	104.3

```
df["bore"] = df["bore"].fillna(df["bore"].median()).astype(float)
df['stroke'] = df["stroke"].fillna(df["bore"].median()).astype(float)
df['horsepower'] = df["horsepower"].fillna(df["bore"].median()).astype(float)
df['peak-rpm'] = df['peak-rpm'].fillna(df['peak-rpm'].median()).astype(float)
df['price'] = df['price'].fillna(df['price'].median()).astype(float)
```

curb-weight	2548	2548	2823	2337	2824	2507	2844	2954	3086	3053	...	3034	2935	3042	3045

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 43 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   symboling               205 non-null    int64  
 1   normalized-losses       205 non-null    int64  
 2   make                    205 non-null    object 
 3   fuel-type               205 non-null    int64  
 4   aspiration              205 non-null    int64  
 5   num-of-doors            205 non-null    float64
 6   drive-wheels            205 non-null    int64  
 7   engine-location         205 non-null    int64  
 8   wheel-base              205 non-null    float64
 9   length                  205 non-null    float64
 10  width                   205 non-null    float64
 11  height                  205 non-null    float64
 12  curb-weight             205 non-null    int64  
 13  num-of-cylinders        205 non-null    int64  
 14  engine-size             205 non-null    int64  
 15  bore                    205 non-null    float64
 16  stroke                  205 non-null    float64
 17  compression-ratio       205 non-null    float64
 18  horsepower              205 non-null    float64
 19  peak-rpm                205 non-null    float64
 20  city-mpg                205 non-null    int64  
 21  highway-mpg             205 non-null    int64  
 22  price                   205 non-null    float64
 23  body-style_convertible  205 non-null    uint8  
 24  body-style_hardtop      205 non-null    uint8  
 25  body-style_hatchback    205 non-null    uint8  
 26  body-style_sedan        205 non-null    uint8  
 27  body-style_wagon        205 non-null    uint8  
 28  engine-type_dohc        205 non-null    uint8  
 29  engine-type_dohcv       205 non-null    uint8  
 30  engine-type_l           205 non-null    uint8  
 31  engine-type_ohc         205 non-null    uint8  
 32  engine-type_ohcf        205 non-null    uint8  
 33  engine-type_ohcv        205 non-null    uint8  
 34  engine-type_rotor       205 non-null    uint8  
 35  fuel-system_1bbl        205 non-null    uint8  
 36  fuel-system_2bbl        205 non-null    uint8  
 37  fuel-system_4bbl        205 non-null    uint8  
 38  fuel-system_idi         205 non-null    uint8  
 39  fuel-system_mfi         205 non-null    uint8  
 40  fuel-system_mpfi        205 non-null    uint8  
 41  fuel-system_spdi        205 non-null    uint8  
 42  fuel-system_spfi        205 non-null    uint8  
dtypes: float64(11), int64(11), object(1), uint8(20)
memory usage: 41.0+ KB
```

fuel-system_spfi	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0

```
# 4 . Fetaure Selection
```

```
df.corr()
```

```
<ipython-input-66-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  df.corr()
```

	symboling	normalized-losses	fuel-type	aspiration	num-of-doors	drive-wheels	engine-location	wheel-base	length	width	...	engine type_ohc
symboling	1.000000	0.457484	-0.194311	-0.059866	-0.663595	-0.111150	0.212471	-0.531954	-0.357612	-0.232919	...	-0.01359
normalized-losses	0.457484	1.000000	-0.104668	-0.011273	-0.348850	0.133824	-0.021510	-0.073709	-0.006837	0.058378	...	0.13071
fuel-type	-0.194311	-0.104668	1.000000	0.401397	0.188496	0.051874	-0.040070	0.308346	0.212679	0.233880	...	-0.08555
aspiration	-0.059866	-0.011273	0.401397	1.000000	0.052803	0.153897	-0.057191	0.257611	0.234539	0.300567	...	-0.07007
drive-wheels	-0.111150	0.133824	0.051874	0.153897	-0.003230	1.000000	0.113823	0.366828	0.416076	0.376554		0.13945

```
# 5. Train Test Split
df = df.select_dtypes(exclude = object) # we drop column having Data type is " object "


df.info() # We fill all NaN Values & Only int and float data type available....Now we ready for further operations
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   symboling               205 non-null    int64
 1   normalized-losses       205 non-null    int64
 2   fuel-type               205 non-null    int64
 3   aspiration              205 non-null    int64
 4   num-of-doors            205 non-null    float64
 5   drive-wheels            205 non-null    int64
 6   engine-location         205 non-null    int64
 7   wheel-base              205 non-null    float64
 8   length                  205 non-null    float64
 9   width                   205 non-null    float64
 10  height                  205 non-null    float64
 11  curb-weight             205 non-null    int64
 12  num-of-cylinders        205 non-null    int64
 13  engine-size             205 non-null    int64
 14  bore                    205 non-null    float64
 15  stroke                  205 non-null    float64
 16  compression-ratio       205 non-null    float64
 17  horsepower              205 non-null    float64
 18  peak-rpm                205 non-null    float64
 19  city-mpg                205 non-null    int64
 20  highway-mpg             205 non-null    int64
 21  price                   205 non-null    float64
 22  body-style_convertible  205 non-null    uint8
 23  body-style_hardtop      205 non-null    uint8
 24  body-style_hatchback    205 non-null    uint8
 25  body-style_sedan        205 non-null    uint8
 26  body-style_wagon        205 non-null    uint8
 27  engine-type_dohc        205 non-null    uint8
 28  engine-type_dohcv       205 non-null    uint8
 29  engine-type_l           205 non-null    uint8
 30  engine-type_ohc         205 non-null    uint8
 31  engine-type_ohcf        205 non-null    uint8
 32  engine-type_ohcv        205 non-null    uint8
 33  engine-type_rotor       205 non-null    uint8
 34  fuel-system_1bbl        205 non-null    uint8
 35  fuel-system_2bbl        205 non-null    uint8
 36  fuel-system_4bbl        205 non-null    uint8
 37  fuel-system_idi         205 non-null    uint8
 38  fuel-system_mfi         205 non-null    uint8
 39  fuel-system_mpfi        205 non-null    uint8
 40  fuel-system_spdi        205 non-null    uint8
 41  fuel-system_spfi        205 non-null    uint8
dtypes: float64(11), int64(11), uint8(20)
memory usage: 39.4 KB
```

type_ohcf	0.057319	-0.210771	-0.092384	-0.054490	0.019837	0.197867	0.453727	-0.185195	-0.116526	-0.124440	...	-0.07511

```
x = df.drop("price",axis = 1) # It returns all Independent Variables
y = df["price"] # It returns all Dependent Variables
```

engine-	0.245950	0.130721	-0.046383	-0.066203	-0.161052	0.131758	-0.017192	-0.081174	-0.057877	-0.013699		-0.03670

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=34)
x_train
```

	symboling	normalized-losses	fuel-type	aspiration	num-of-doors	drive-wheels	engine-location	wheel-base	length	width	...	engine-type_ohcv	engine-type_rotor	fuel-system_1bbl	s
203	-1	95	1	1	4.0	1	0	109.1	188.8	68.9	...	0	0	0	
204	-1	95	0	1	4.0	1	0	109.1	188.8	68.9	...	0	0	0	
52	1	104	0	0	2.0	0	0	93.1	159.1	64.2	...	0	0	0	
2	1	115	0	0	2.0	1	0	94.5	171.2	65.5	...	1	0	0	
46	2	115	0	0	2.0	1	0	96.0	172.6	65.2	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
68	-1	93	1	1	4.0	1	0	110.0	190.9	70.3	...	0	0	0	
85	1	125	0	0	4.0	0	0	96.3	172.4	65.4	...	0	0	0	
105	3	194	0	1	2.0	1	0	91.3	170.7	67.9	...	1	0	0	
122	1	154	0	0	4.0	0	0	93.7	167.3	63.8	...	0	0	0	
161	0	91	0	0	4.0	0	0	95.7	166.3	64.4	...	0	0	0	

164 rows × 41 columns

```
x_train.shape
```

```
(164, 41)
```

```
# 6. Model Training
```

```
model = LinearRegression()
model.fit(x_train,y_train)
```

```
LinearRegression()
```

```
 # 7. Model Evaluation
 y_pred_test = model.predict(x_test) # To find Predicted Values
```

```
#Testing data evalution
y_pred = model.predict(x_test)
y_pred[20:25]
```

```
array([11100.00500014, 19909.19320479, 10866.041479  ,  7933.8058285 ,
        7669.28848162])
```

```
y_pred_test[20:25] # Predicted values
```

```
array([11100.00500014, 19909.19320479, 10866.041479  ,  7933.8058285 ,
        7669.28848162])
```

```
y_test[20:25] # actual values
```

```
123     8921.0
178    16558.0
3      13950.0
76      5389.0
79      7689.0
Name: price, dtype: float64
```

```
# 8. Testing Data Evalution
```

```
mse = mean_squared_error(y_test,y_pred_test)
print("MSE is:",mse)
rmse = np.sqrt(mse)
print("RMSE:",rmse)
mae = mean_absolute_error(y_test,y_pred_test)
print("MAE is:",mae)
r2 = r2_score(y_test,y_pred_test)
```

```
print("r2 score is :",r2)
```

```
MSE is: 14154182.910287894
RMSE: 3762.204527971319
MAE is: 2392.8348790513933
r2 score is : 0.7301625587922318
```

```
# 9. Training Data Evalution
y_pred_train = model.predict(x_train)
mse = mean_squared_error(y_train,y_pred_train)
print("MSE is:",mse)
rmse = np.sqrt(mse)
print("RMSE:",rmse)
mae = mean_absolute_error(y_train,y_pred_train)
print("MAE is:",mae)
r2 = r2_score(y_train,y_pred_train)
print("r2 score is :",r2)
```

```
MSE is: 4974235.837890235
RMSE: 2230.299495110519
MAE is: 1585.3408437358958
r2 score is : 0.9223636932435281
```