car-price-prediction

December 10, 2023

Car Price Prediction

Importing libraries and dataset

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/CarPrice_Assignment (1).csv')
```

Data Preprocessing

[ ]:
```
df.head()
```

[ ]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	\
0	1	3	alfa-romero giulia	gas	std	two	
1	2	3	alfa-romero stelvio	gas	std	two	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	
3	4	2	audi 100 ls	gas	std	four	
4	5	2	audi 100ls	gas	std	four	

	carbody	drivewheel	enginelocation	wheelbase	…	enginesize	\
0	convertible	rwd	front	88.6	…	130	
1	convertible	rwd	front	88.6	…	130	
2	hatchback	rwd	front	94.5	…	152	
3	sedan	fwd	front	99.8	…	109	
4	sedan	4wd	front	99.4	…	136	

	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	\
0	mpfi	3.47	2.68	9.0	111	5000	21	
1	mpfi	3.47	2.68	9.0	111	5000	21	
2	mpfi	2.68	3.47	9.0	154	5000	19	
3	mpfi	3.19	3.40	10.0	102	5500	24	
4	mpfi	3.19	3.40	8.0	115	5500	18	

	highwaympg	price
0	27	13495.0
1	27	16500.0
2	26	16500.0

```
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

[ ]:
```
df.tail()
```

[ ]:
```
     car_ID  symboling          CarName fueltype aspiration doornumber  \
200     201         -1  volvo 145e (sw)      gas        std       four
201     202         -1      volvo 144ea      gas      turbo       four
202     203         -1      volvo 244dl      gas        std       four
203     204         -1        volvo 246   diesel      turbo       four
204     205         -1      volvo 264gl      gas      turbo       four

    carbody drivewheel enginelocation  wheelbase  …  enginesize  fuelsystem  \
200   sedan        rwd          front      109.1  …         141        mpfi
201   sedan        rwd          front      109.1  …         141        mpfi
202   sedan        rwd          front      109.1  …         173        mpfi
203   sedan        rwd          front      109.1  …         145         idi
204   sedan        rwd          front      109.1  …         141        mpfi

     boreratio  stroke compressionratio horsepower  peakrpm citympg  \
200       3.78    3.15              9.5        114     5400      23
201       3.78    3.15              8.7        160     5300      19
202       3.58    2.87              8.8        134     5500      18
203       3.01    3.40             23.0        106     4800      26
204       3.78    3.15              9.5        114     5400      19

     highwaympg    price
200          28  16845.0
201          25  19045.0
202          23  21485.0
203          27  22470.0
204          25  22625.0

[5 rows x 26 columns]
```

[ ]:
```
df.dtypes
```

[ ]:
```
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
```

[ ]:
```
df.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
```

[ ]:
```
df.columns
```

[ ]:
```
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')
```

[ ]:
```
df.describe()
```

[ ]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000

	curbweight	enginesize	boreratio	stroke	compressionratio
count	205.000000	205.000000	205.000000	205.000000	205.000000
mean	2555.565854	126.907317	3.329756	3.255415	10.142537
std	520.680204	41.642693	0.270844	0.313597	3.972040
min	1488.000000	61.000000	2.540000	2.070000	7.000000
25%	2145.000000	97.000000	3.150000	3.110000	8.600000
50%	2414.000000	120.000000	3.310000	3.290000	9.000000
75%	2935.000000	141.000000	3.580000	3.410000	9.400000
max	4066.000000	326.000000	3.940000	4.170000	23.000000

	horsepower	peakrpm	citympg	highwaympg	price
count	205.000000	205.000000	205.000000	205.000000	205.000000
mean	104.117073	5125.121951	25.219512	30.751220	13276.710571
std	39.544167	476.985643	6.542142	6.886443	7988.852332
min	48.000000	4150.000000	13.000000	16.000000	5118.000000
25%	70.000000	4800.000000	19.000000	25.000000	7788.000000

```
50%     95.000000  5200.000000   24.000000   30.000000  10295.000000
75%    116.000000  5500.000000   30.000000   34.000000  16503.000000
max    288.000000  6600.000000   49.000000   54.000000  45400.000000
```

[ ]:
```
df.isna().sum()
```

[ ]:
```
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
```

[ ]:
```
df.drop(['car_ID','CarName'],axis=1,inplace=True)
df
```

[ ]:
```
     symboling fueltype aspiration doornumber      carbody drivewheel  \
0            3      gas        std        two  convertible        rwd
1            3      gas        std        two  convertible        rwd
2            1      gas        std        two    hatchback        rwd
3            2      gas        std       four        sedan        fwd
4            2      gas        std       four        sedan        4wd
..         …      …        …         …          …        …
200         -1      gas        std       four        sedan        rwd
201         -1      gas      turbo       four        sedan        rwd
202         -1      gas        std       four        sedan        rwd
```

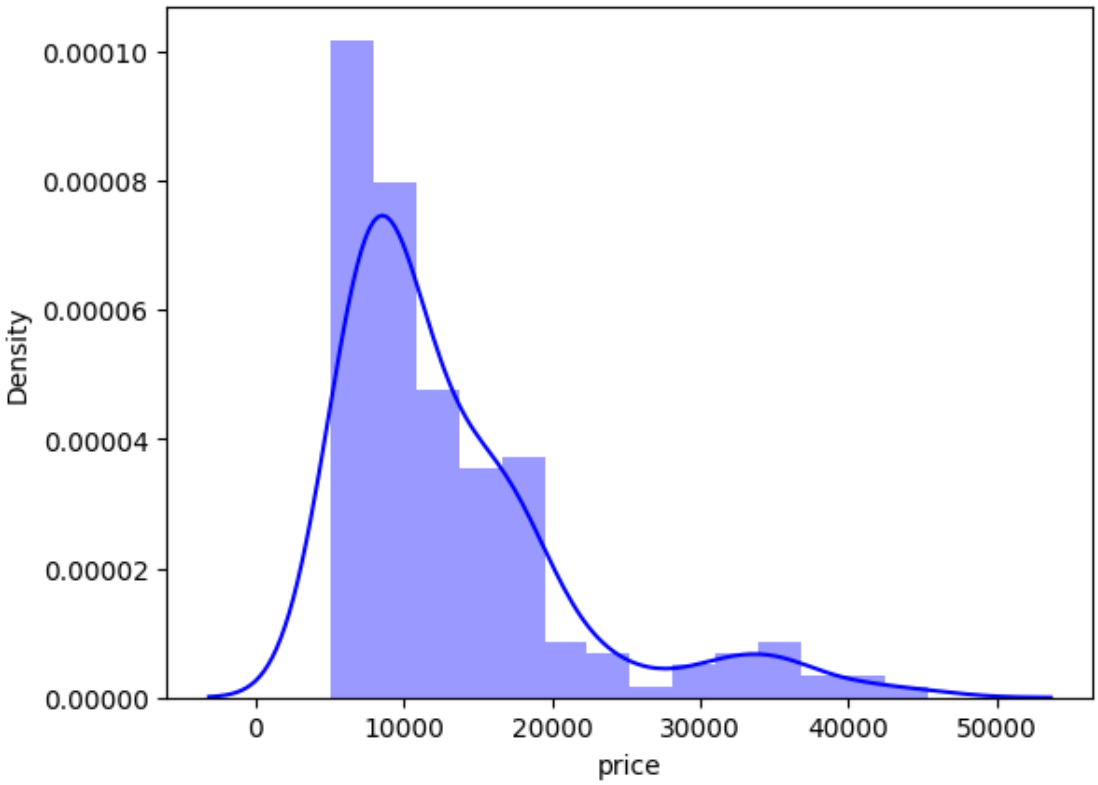
```
203         -1   diesel      turbo       four        sedan        rwd
204         -1      gas      turbo       four        sedan        rwd

    enginelocation  wheelbase  carlength  carwidth  …  enginesize  \
0            front       88.6      168.8      64.1  …         130
1            front       88.6      168.8      64.1  …         130
2            front       94.5      171.2      65.5  …         152
3            front       99.8      176.6      66.2  …         109
4            front       99.4      176.6      66.4  …         136
..             …          …         …        …  …           …
200          front      109.1      188.8      68.9  …         141
201          front      109.1      188.8      68.8  …         141
202          front      109.1      188.8      68.9  …         173
203          front      109.1      188.8      68.9  …         145
204          front      109.1      188.8      68.9  …         141

     fuelsystem boreratio stroke  compressionratio horsepower  peakrpm  \
0          mpfi      3.47   2.68               9.0        111     5000
1          mpfi      3.47   2.68               9.0        111     5000
2          mpfi      2.68   3.47               9.0        154     5000
3          mpfi      3.19   3.40              10.0        102     5500
4          mpfi      3.19   3.40               8.0        115     5500
..          …         …     …                 …          …       …
200        mpfi      3.78   3.15               9.5        114     5400
201        mpfi      3.78   3.15               8.7        160     5300
202        mpfi      3.58   2.87               8.8        134     5500
203         idi      3.01   3.40              23.0        106     4800
204        mpfi      3.78   3.15               9.5        114     5400

     citympg  highwaympg    price
0         21          27  13495.0
1         21          27  16500.0
2         19          26  16500.0
3         24          30  13950.0
4         18          22  17450.0
..       …           …      …
200       23          28  16845.0
201       19          25  19045.0
202       18          23  21485.0
203       26          27  22470.0
204       19          25  22625.0

[205 rows x 24 columns]
```

Data Visualization

[ ]:
```
sns.distplot(df['price'],color='blue')
```

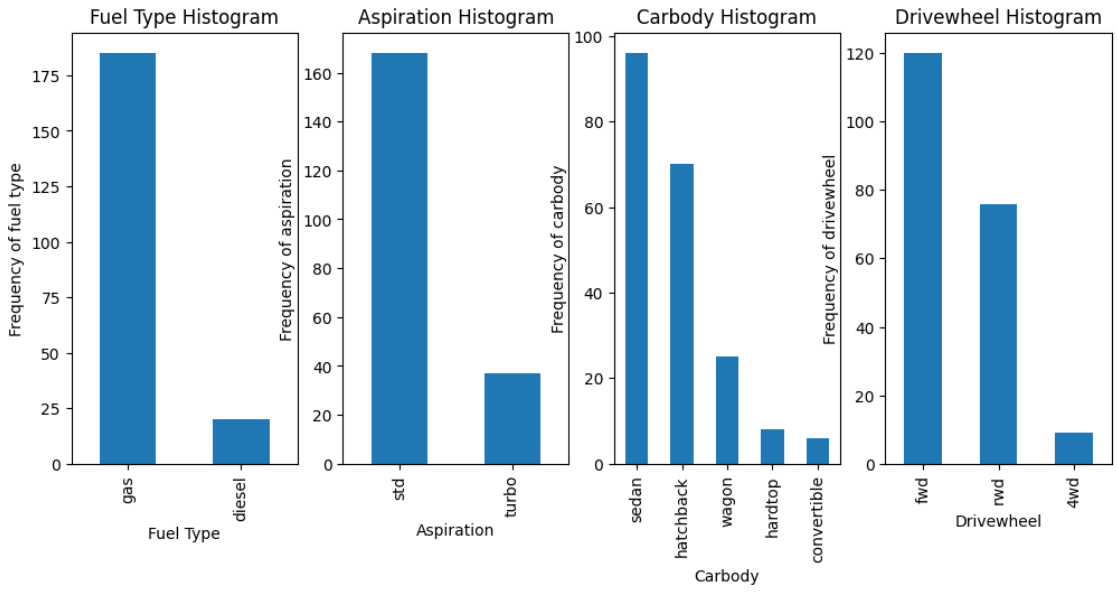
```
<ipython-input-10-3c1e2e7c2d5e>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['price'],color='blue')
```

[ ]: <Axes: xlabel='price', ylabel='Density'>



[ ]:
```
plt.figure(figsize=(12,5))

plt.subplot(1,4,1)
plt1=df.fueltype.value_counts().plot(kind='bar')
plt.title('Fuel Type Histogram')
plt.xlabel('Fuel Type')
plt.ylabel('Frequency of fuel type')
```

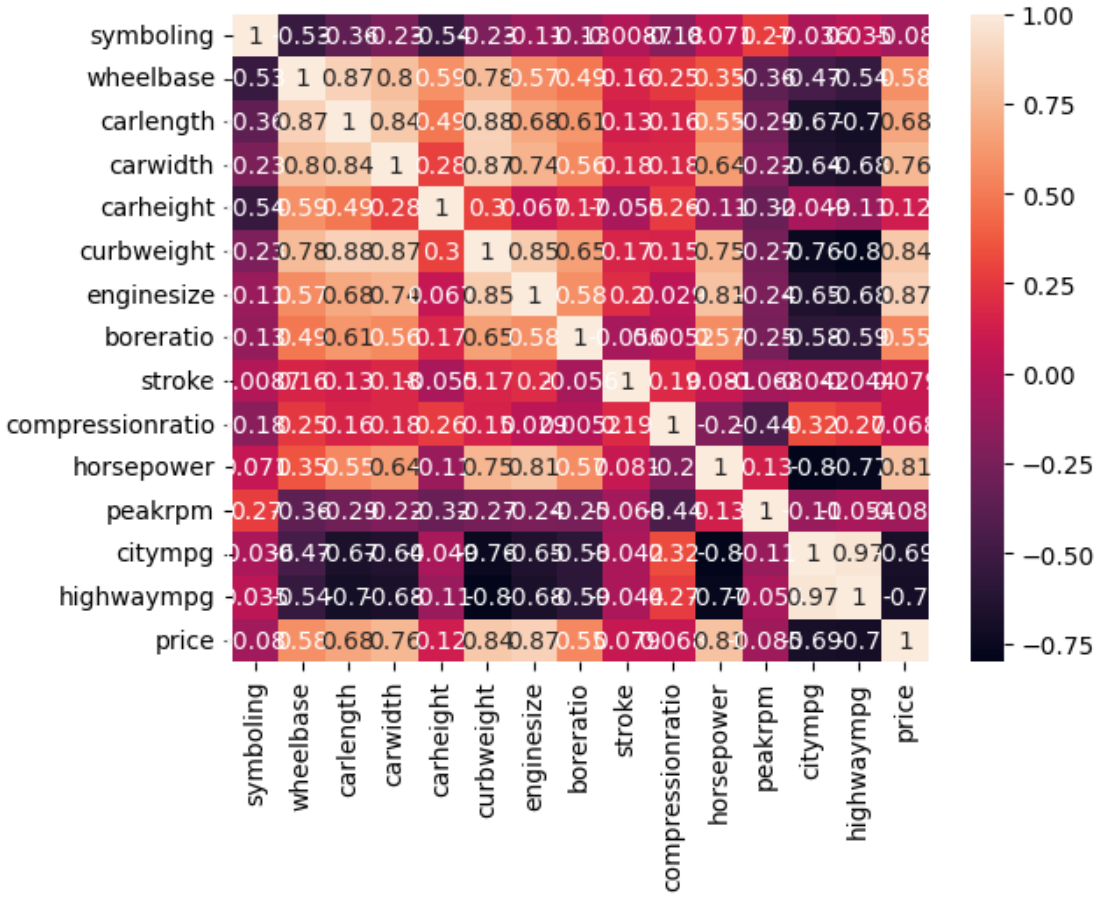
```
plt.subplot(1,4,2)
plt2=df.aspiration.value_counts().plot(kind='bar')
plt.title('Aspiration Histogram')
plt.xlabel('Aspiration')
plt.ylabel('Frequency of aspiration')

plt.subplot(1,4,3)
plt3=df.carbody.value_counts().plot(kind='bar')
plt.title('Carbody Histogram')
plt.xlabel('Carbody')
plt.ylabel('Frequency of carbody')


plt.subplot(1,4,4)
plt4=df.drivewheel.value_counts().plot(kind='bar')
plt.title('Drivewheel Histogram')
plt.xlabel('Drivewheel')
plt.ylabel('Frequency of drivewheel')

plt.show()
```



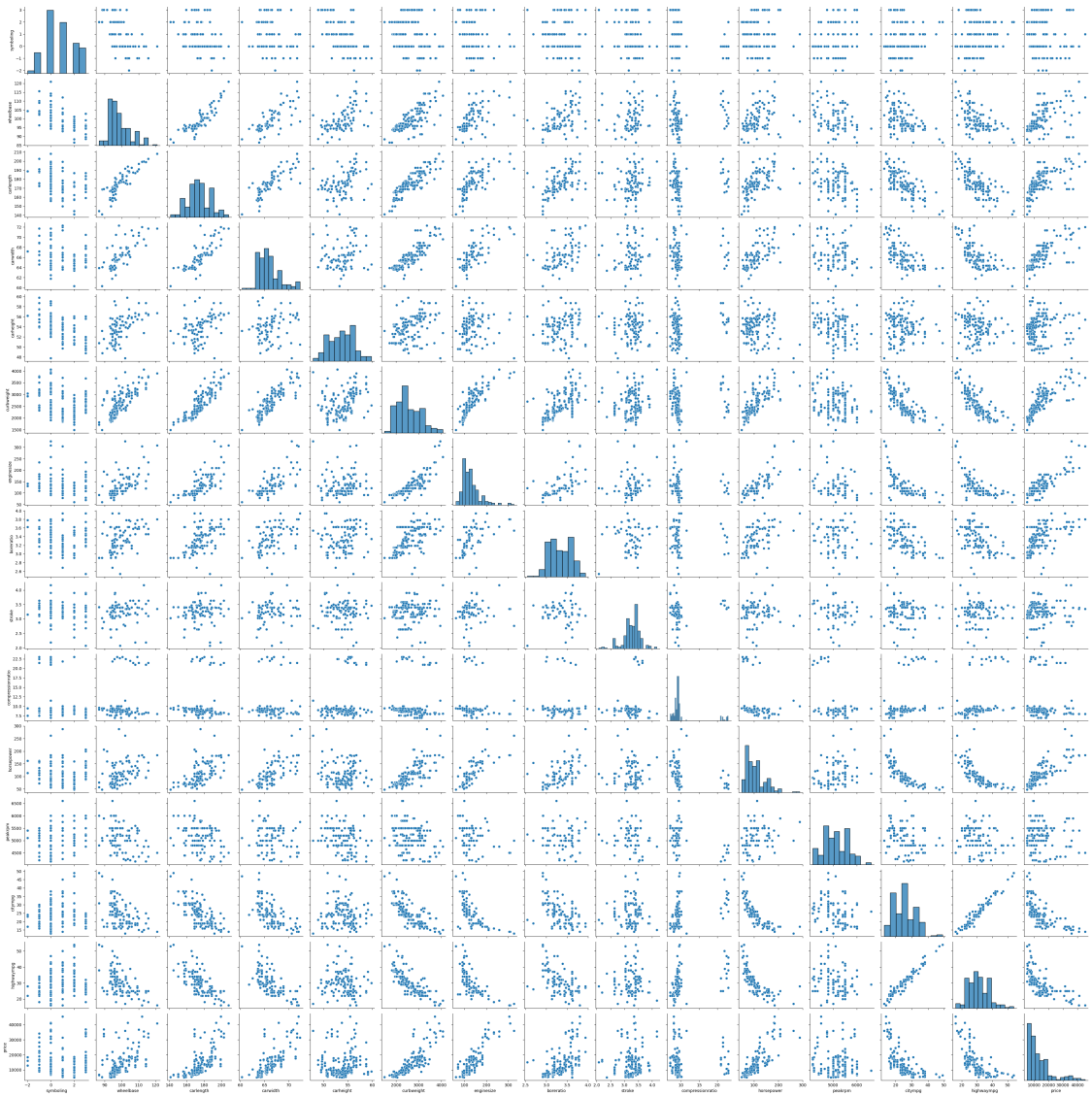
[ ]:
```
sns.heatmap(df.corr(),annot=True)
plt.show()
```

```
<ipython-input-12-f6412ee67fb3>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
```

```
sns.heatmap(df.corr(),annot=True)
```



[ ]:
```
sns.pairplot(df)
plt.show()
```



Encoding using Label Encoding

[ ]:
```
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
df['fueltype']=label.fit_transform(df['fueltype'])
df['aspiration']=label.fit_transform(df['aspiration'])
df['carbody']=label.fit_transform(df['carbody'])
df['doornumber']=label.fit_transform(df['doornumber'])
df['drivewheel']=label.fit_transform(df['drivewheel'])
df['enginelocation']=label.fit_transform(df['enginelocation'])
df['enginetype']=label.fit_transform(df['enginetype'])
df['fuelsystem']=label.fit_transform(df['fuelsystem'])
df
```

[ ]:

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	\
0	3	1	0	1	0	2	
1	3	1	0	1	0	2	
2	1	1	0	1	2	2	
3	2	1	0	0	3	1	
4	2	1	0	0	3	0	
..	…	…	…	…	…	…	
200	-1	1	0	0	3	2	
201	-1	1	1	0	3	2	
202	-1	1	0	0	3	2	
203	-1	0	1	0	3	2	
204	-1	1	1	0	3	2	

	enginelocation	wheelbase	carlength	carwidth	…	enginesize	\
0	0	88.6	168.8	64.1	…	130	
1	0	88.6	168.8	64.1	…	130	
2	0	94.5	171.2	65.5	…	152	
3	0	99.8	176.6	66.2	…	109	
4	0	99.4	176.6	66.4	…	136	
..	…	…	…	…	…	…	
200	0	109.1	188.8	68.9	…	141	
201	0	109.1	188.8	68.8	…	141	
202	0	109.1	188.8	68.9	…	173	
203	0	109.1	188.8	68.9	…	145	
204	0	109.1	188.8	68.9	…	141	

	fuelsystem	boreratio	stroke	compressionratio	horsepower	peakrpm	\
0	5	3.47	2.68	9.0	111	5000	
1	5	3.47	2.68	9.0	111	5000	
2	5	2.68	3.47	9.0	154	5000	
3	5	3.19	3.40	10.0	102	5500	
4	5	3.19	3.40	8.0	115	5500	
..	…	…	…	…	…	…	
200	5	3.78	3.15	9.5	114	5400	
201	5	3.78	3.15	8.7	160	5300	
202	5	3.58	2.87	8.8	134	5500	
203	3	3.01	3.40	23.0	106	4800	
204	5	3.78	3.15	9.5	114	5400	

	citympg	highwaympg	price
0	21	27	13495.0
1	21	27	16500.0
2	19	26	16500.0
3	24	30	13950.0
4	18	22	17450.0
..	…	…	…
200	23	28	16845.0

```
201        19           25  19045.0
202        18           23  21485.0
203        26           27  22470.0
204        19           25  22625.0

[205 rows x 24 columns]
```

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

Input and output Separation

[ ]:
```
x=df.iloc[:,:-1].values
x
```

[ ]:
```
array([[3, 1, 0, …, 5000, 21, 27],
       [3, 1, 0, …, 5000, 21, 27],
       [1, 1, 0, …, 5000, 19, 26],
       …,
       [-1, 1, 0, …, 5500, 18, 23],
       [-1, 0, 1, …, 4800, 26, 27],
       [-1, 1, 1, …, 5400, 19, 25]], dtype=object)
```

[ ]:
```
y=df.iloc[:,-1].values
y
```

[ ]:
```
array([13495.   , 16500.   , 16500.   , 13950.   , 17450.   , 15250.   ,
       17710.   , 18920.   , 23875.   , 17859.167, 16430.   , 16925.   ,
       20970.   , 21105.   , 24565.   , 30760.   , 41315.   , 36880.   ,
        5151.   ,  6295.   ,  6575.   ,  5572.   ,  6377.   ,  7957.   ,
        6229.   ,  6692.   ,  7609.   ,  8558.   ,  8921.   , 12964.   ,
        6479.   ,  6855.   ,  5399.   ,  6529.   ,  7129.   ,  7295.   ,
        7295.   ,  7895.   ,  9095.   ,  8845.   , 10295.   , 12945.   ,
       10345.   ,  6785.   ,  8916.5  ,  8916.5  , 11048.   , 32250.   ,
       35550.   , 36000.   ,  5195.   ,  6095.   ,  6795.   ,  6695.   ,
        7395.   , 10945.   , 11845.   , 13645.   , 15645.   ,  8845.   ,
        8495.   , 10595.   , 10245.   , 10795.   , 11245.   , 18280.   ,
       18344.   , 25552.   , 28248.   , 28176.   , 31600.   , 34184.   ,
       35056.   , 40960.   , 45400.   , 16503.   ,  5389.   ,  6189.   ,
        6669.   ,  7689.   ,  9959.   ,  8499.   , 12629.   , 14869.   ,
       14489.   ,  6989.   ,  8189.   ,  9279.   ,  9279.   ,  5499.   ,
        7099.   ,  6649.   ,  6849.   ,  7349.   ,  7299.   ,  7799.   ,
        7499.   ,  7999.   ,  8249.   ,  8949.   ,  9549.   , 13499.   ,
       14399.   , 13499.   , 17199.   , 19699.   , 18399.   , 11900.   ,
       13200.   , 12440.   , 13860.   , 15580.   , 16900.   , 16695.   ,
       17075.   , 16630.   , 17950.   , 18150.   ,  5572.   ,  7957.   ,
        6229.   ,  6692.   ,  7609.   ,  8921.   , 12764.   , 22018.   ,
```

```
       32528.   , 34028.   , 37028.   , 31400.5  ,  9295.   ,  9895.   ,
       11850.   , 12170.   , 15040.   , 15510.   , 18150.   , 18620.   ,
        5118.   ,  7053.   ,  7603.   ,  7126.   ,  7775.   ,  9960.   ,
        9233.   , 11259.   ,  7463.   , 10198.   ,  8013.   , 11694.   ,
        5348.   ,  6338.   ,  6488.   ,  6918.   ,  7898.   ,  8778.   ,
        6938.   ,  7198.   ,  7898.   ,  7788.   ,  7738.   ,  8358.   ,
        9258.   ,  8058.   ,  8238.   ,  9298.   ,  9538.   ,  8449.   ,
        9639.   ,  9989.   , 11199.   , 11549.   , 17669.   ,  8948.   ,
       10698.   ,  9988.   , 10898.   , 11248.   , 16558.   , 15998.   ,
       15690.   , 15750.   ,  7775.   ,  7975.   ,  7995.   ,  8195.   ,
        8495.   ,  9495.   ,  9995.   , 11595.   ,  9980.   , 13295.   ,
       13845.   , 12290.   , 12940.   , 13415.   , 15985.   , 16515.   ,
       18420.   , 18950.   , 16845.   , 19045.   , 21485.   , 22470.   ,
       22625.   ])
```

Splitting the training and testing data

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=0.
↪30)
```

[ ]:
```
x_train
```

[ ]:
```
array([[-1, 1, 0, …, 4200, 27, 32],
       [1, 1, 1, …, 5000, 19, 24],
       [-1, 0, 1, …, 4500, 30, 33],
       …,
       [1, 1, 0, …, 5200, 31, 37],
       [3, 1, 0, …, 5200, 19, 24],
       [0, 1, 0, …, 5200, 17, 22]], dtype=object)
```

[ ]:
```
x_test
```

[ ]:
```
array([[0, 1, 0, …, 5400, 16, 22],
       [0, 1, 1, …, 5500, 16, 22],
       [0, 1, 0, …, 5200, 27, 34],
       …,
       [2, 1, 0, …, 5500, 37, 41],
       [0, 1, 0, …, 4800, 24, 25],
       [3, 1, 0, …, 5200, 19, 25]], dtype=object)
```

[ ]:
```
y_train
```

[ ]:
```
array([11248. , 16503. , 10698. ,  6855. , 20970. ,  7295. , 22018. ,
       10345. ,  6938. , 18150. ,  7603. ,  6095. , 16695. ,  9538. ,
       12945. , 12764. ,  6669. ,  7957. ,  7609. , 18620. , 16845. ,
       18150. , 16500. ,  6189. , 11048. ,  7609. , 34028. ,  6989. ,
```

```
 5118. ,  7295. , 17075. , 10595. ,  6377. , 15510. , 13845. ,
 6529. , 16925. , 18950. ,  8058. , 17710. ,  8558. , 32528. ,
15985. ,  7126. , 17450. ,  5399. , 12440. ,  9495. , 16630. ,
 8013. , 16430. , 10245. ,  5572. ,  5348. , 11199. , 13495. ,
 7995. ,  7975. , 31600. , 22470. , 11245. ,  8916.5,  9298. ,
 8921. , 10295. , 13200. ,  8921. ,  7957. ,  8778. , 11595. ,
 8499. ,  8845. , 17669. , 32250. ,  7299. ,  7898. ,  6785. ,
 7775. , 13950. , 19699. ,  6695. , 12170. ,  8195. ,  8358. ,
 9980. , 36000. ,  9959. ,  7129. , 18920. , 13860. ,  6649. ,
14869. , 15750. ,  8495. ,  5499. , 23875. , 21105. ,  8845. ,
10898. ,  9895. , 36880. , 35056. , 15690. , 15040. , 11549. ,
18420. , 10795. ,  7395. , 11900. ,  5195. , 19045. ,  9989. ,
15645. , 35550. ,  9279. ,  5572. , 13645. ,  7738. , 16515. ,
31400.5,  7895. ,  7198. , 12290. , 16500. ,  6795. , 11694. ,
 9295. ,  6338. , 13499. ,  8949. , 17950. ,  9279. , 45400. ,
 6692. , 22625. ,  6575. ,  9995. , 34184. , 18399. , 24565. ,
 6849. , 15998. , 14399. ])
```

[ ]:
```
y_test
```

[ ]:
```
array([30760.   , 17859.167,  9549.   , 11850.   , 28248.   ,  7799.   ,
        7788.   ,  9258.   , 10198.   ,  7775.   , 13295.   ,  8238.   ,
       18280.   ,  9988.   , 40960.   ,  6488.   ,  5151.   , 12629.   ,
        8189.   ,  9960.   ,  8495.   , 13499.   ,  8249.   ,  6479.   ,
        6692.   , 41315.   ,  9639.   , 13415.   ,  7999.   , 12940.   ,
       25552.   ,  6229.   ,  7898.   , 21485.   ,  7689.   , 28176.   ,
       11259.   , 10945.   ,  8916.5  , 14489.   ,  7463.   , 18344.   ,
       15580.   ,  6918.   ,  7499.   ,  9095.   ,  6229.   ,  7053.   ,
       16900.   , 12964.   ,  6295.   , 16558.   ,  7099.   ,  8948.   ,
        7349.   , 15250.   ,  8449.   , 11845.   , 37028.   ,  5389.   ,
        9233.   , 17199.   ])
```

Normalization/Scaling

[ ]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

[ ]:
```
x_train
```

[ ]:
```
array([[-1.5       ,  0.30265996, -0.48245064, …, -1.96616697,
         0.37011796,  0.27024799],
       [ 0.125     ,  0.30265996,  2.0727509 , …, -0.31833179,
        -0.98698122, -1.01793411],
       [-1.5       , -3.30403793,  2.0727509 , …, -1.34822878,
```

```
         0.87903015,  0.43127076],
       …,
       [ 0.125     ,  0.30265996, -0.48245064, …,  0.093627  ,
         1.04866755,  1.07536181],
       [ 1.75      ,  0.30265996, -0.48245064, …,  0.093627  ,
        -0.98698122, -1.01793411],
       [-0.6875    ,  0.30265996, -0.48245064, …,  0.093627  ,
        -1.32625602, -1.33997964]])
```

[ ]:
```
x_test
```

[ ]:
```
array([[-0.63602731,  0.38490018, -0.43852901, …,  0.76952735,
        -1.31581608, -1.20210967],
       [-0.63602731,  0.38490018,  2.28035085, …,  0.99404827,
        -1.31581608, -1.20210967],
       [-0.63602731,  0.38490018, -0.43852901, …,  0.32048551,
         0.11087163,  0.27863469],
       …,
       [ 0.94132042,  0.38490018, -0.43852901, …,  0.99404827,
         1.40786045,  1.14240223],
       [-0.63602731,  0.38490018, -0.43852901, …, -0.57759818,
        -0.27822502, -0.83192358],
       [ 1.72999429,  0.38490018, -0.43852901, …,  0.32048551,
        -0.92671943, -0.83192358]])
```

Model Creation

[ ]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
y_pred=lr.predict(x_test)
y_pred
```

[ ]:
```
array([24903.24683842, 16619.47172704, 10832.67821138, 13763.56497872,
       25353.79014644,  6835.79340343,  8628.66263625,  8157.10879881,
       10114.11669516,  7551.12822609, 16300.84425984,  8056.74071576,
       16479.17022752, 11910.72725318, 40443.4444885 ,  8765.15605235,
        -694.53216509, 15625.16342628, 10918.28206209, 11024.4443142 ,
       11937.99431802, 21106.50121293,  8529.41333967,  3769.60396739,
        7899.89650122, 23648.6358909 , 15180.56004204, 16922.09914644,
        5953.47962008, 17437.90325683, 25780.94334098,  9008.09177933,
        5928.72018172, 24801.45825054,  9170.03892285, 26065.23365799,
       10850.36585588, 11390.86469163,  6963.62691685, 15736.34285506,
        8907.17268595, 13538.46612218, 20119.20981918,  6492.04735498,
```

```
 7565.44679054, 10754.25701687,  9008.09177933,  8202.45888483,
19155.49099848, 15524.34244375,  6544.9542917 , 21154.48411189,
 4249.21758089, 10540.69634271,  5933.84496233, 14027.70774798,
15186.60147519, 11390.86469163, 33794.95563831,  7498.15686774,
 9951.76019171, 22116.21294033])
```

[ ]:
```
from sklearn import metrics
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
mae=mean_absolute_error(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
rmse=np.sqrt(mse)
```

[ ]:
```
print('mean absolute error',mae)
print('mean squared error',mse)
print('r2_score',r2)
print('root mean squared error',rmse)
```

```
mean absolute error 2499.49008388361
mean squared error 13044156.963046974
r2_score 0.8117301876050342
root mean squared error 3611.6695534125174
```