In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
dataset=pd.read_csv(r'FuelConsumption.csv')
dataset
```

Out[2]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSIC
0	2014	ACURA	ILX	COMPACT	2.0	4	A
1	2014	ACURA	ILX	COMPACT	2.4	4	M
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	A
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	A
...	...	...	...	...	...	...	
1062	2014	VOLVO	XC60 AWD	SUV - SMALL	3.0	6	A
1063	2014	VOLVO	XC60 AWD	SUV - SMALL	3.2	6	A
1064	2014	VOLVO	XC70 AWD	SUV - SMALL	3.0	6	A
1065	2014	VOLVO	XC70 AWD	SUV - SMALL	3.2	6	A
1066	2014	VOLVO	XC90 AWD	SUV - STANDARD	3.2	6	A

1067 rows × 13 columns

In [3]:

```
type(dataset)
```

Out[3]:

```
pandas.core.frame.DataFrame
```

In [4]:

```
dataset.drop(['MODELYEAR','MAKE','MODEL','VEHICLECLASS','TRANSMISSION','FUELCONSUMPTION_COM
dataset
```

Out[4]:

	ENGINESIZE	CYLINDERS	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HW
0	2.0	4	Z	9.9	6
1	2.4	4	Z	11.2	7
2	1.5	4	Z	6.0	5
3	3.5	6	Z	12.7	9
4	3.5	6	Z	12.1	8
...	...	...	...	...	
1062	3.0	6	X	13.4	9
1063	3.2	6	X	13.2	9
1064	3.0	6	X	13.4	9
1065	3.2	6	X	12.9	9
1066	3.2	6	X	14.9	10

1067 rows × 7 columns

In [5]:

```
dataset.isnull().any()
```

Out[5]:

```
ENGINESIZE              False
CYLINDERS               False
FUELTYPE                False
FUELCONSUMPTION_CITY    False
FUELCONSUMPTION_HWY     False
FUELCONSUMPTION_COMB    False
CO2EMISSIONS            False
dtype: bool
```

In [6]:

```
dataset.isnull().sum()
```

Out[6]:

```
ENGINESIZE              0
CYLINDERS               0
FUELTYPE                0
FUELCONSUMPTION_CITY    0
FUELCONSUMPTION_HWY     0
FUELCONSUMPTION_COMB    0
CO2EMISSIONS            0
dtype: int64
```

In [7]:

```
dataset.corr()
```

Out[7]:

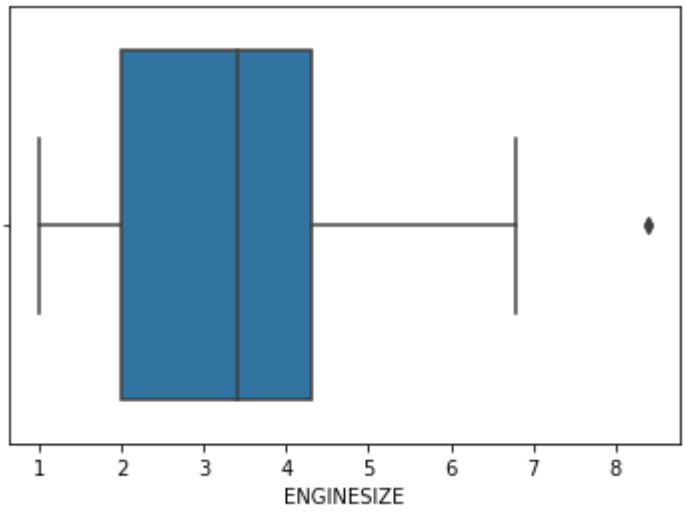
	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUM
ENGINESIZE	1.000000	0.934011	0.832225	
CYLINDERS	0.934011	1.000000	0.796473	
FUELCONSUMPTION_CITY	0.832225	0.796473	1.000000	
FUELCONSUMPTION_HWY	0.778746	0.724594	0.965718	
FUELCONSUMPTION_COMB	0.819482	0.776788	0.995542	
CO2EMISSIONS	0.874154	0.849685	0.898039	

In [8]:

```
import seaborn as sns
sns.boxplot(dataset['ENGINESIZE'])
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28c8af2c548>
```

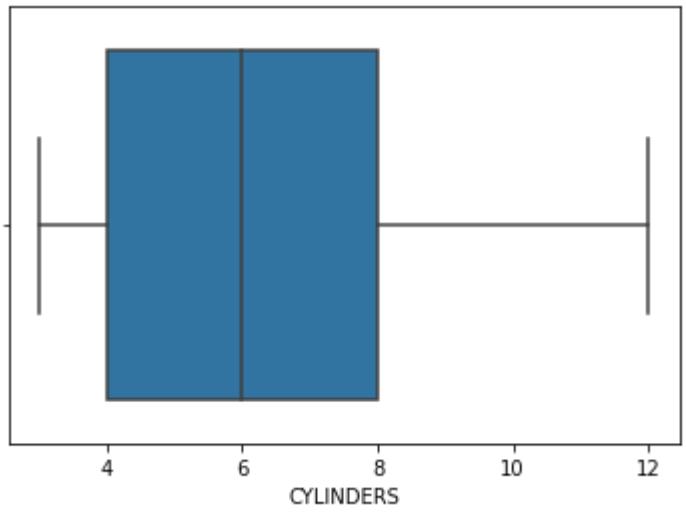


In [9]:

```
sns.boxplot(dataset['CYLINDERS'])
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28c8c663188>
```

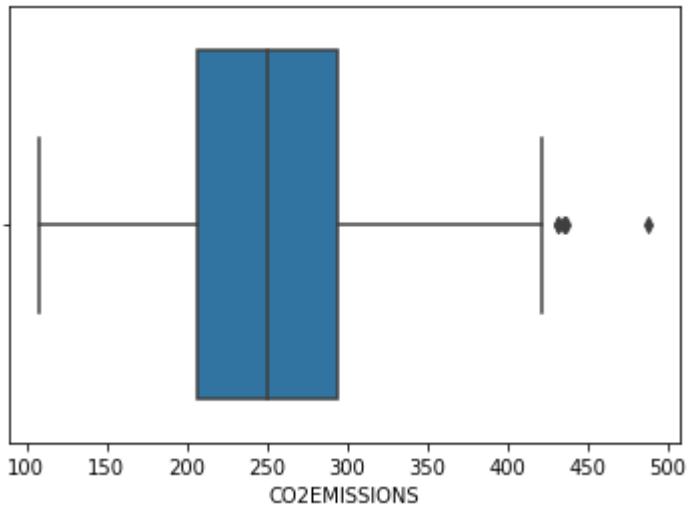


In [10]:

```
sns.boxplot(dataset['CO2EMISSIONS'])
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28c8c6d7908>
```

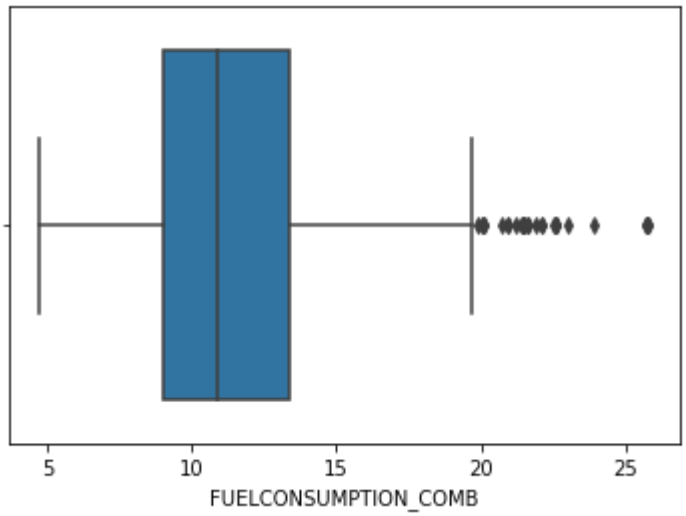


In [11]:

```
sns.boxplot(dataset['FUELCONSUMPTION_COMB'])
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x28c8c73d888>
```



In [12]:

```
x=dataset.iloc[:,0:6].values
x
```

Out[12]:

```
array([[2.0, 4, 'Z', 9.9, 6.7, 8.5],
       [2.4, 4, 'Z', 11.2, 7.7, 9.6],
       [1.5, 4, 'Z', 6.0, 5.8, 5.9],
       ...,
       [3.0, 6, 'X', 13.4, 9.8, 11.8],
       [3.2, 6, 'X', 12.9, 9.3, 11.3],
       [3.2, 6, 'X', 14.9, 10.2, 12.8]], dtype=object)
```

In [13]:

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

Out[13]:

```
array([196, 221, 136, ..., 271, 260, 294], dtype=int64)
```

In [14]:

```
dataset['FUELTYPE'].value_counts()
```

Out[14]:

```
X    514
Z    434
E     92
D     27
Name: FUELTYPE, dtype: int64
```

In [15]:

```
import sklearn
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

In [16]:

```
ct=ColumnTransformer([("on",OneHotEncoder(),[2])],remainder='passthrough')
x=ct.fit_transform(x)
x
```

Out[16]:

```
array([[0.0, 0.0, 0.0, ..., 9.9, 6.7, 8.5],
       [0.0, 0.0, 0.0, ..., 11.2, 7.7, 9.6],
       [0.0, 0.0, 0.0, ..., 6.0, 5.8, 5.9],
       ...,
       [0.0, 0.0, 1.0, ..., 13.4, 9.8, 11.8],
       [0.0, 0.0, 1.0, ..., 12.9, 9.3, 11.3],
       [0.0, 0.0, 1.0, ..., 14.9, 10.2, 12.8]], dtype=object)
```

In [17]:

```
x=x[:,1:]
x
```

Out[17]:

```
array([[0.0, 0.0, 1.0, ..., 9.9, 6.7, 8.5],
       [0.0, 0.0, 1.0, ..., 11.2, 7.7, 9.6],
       [0.0, 0.0, 1.0, ..., 6.0, 5.8, 5.9],
       ...,
       [0.0, 1.0, 0.0, ..., 13.4, 9.8, 11.8],
       [0.0, 1.0, 0.0, ..., 12.9, 9.3, 11.3],
       [0.0, 1.0, 0.0, ..., 14.9, 10.2, 12.8]], dtype=object)
```

In [18]:

```
x.shape
```

Out[18]:

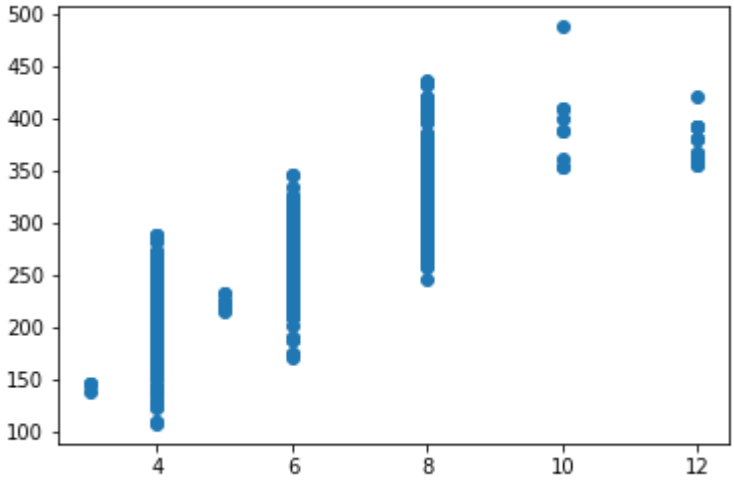
```
(1067, 8)
```

In [19]:

```
plt.scatter(x[:,4],y)
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x28c8c960d48>
```

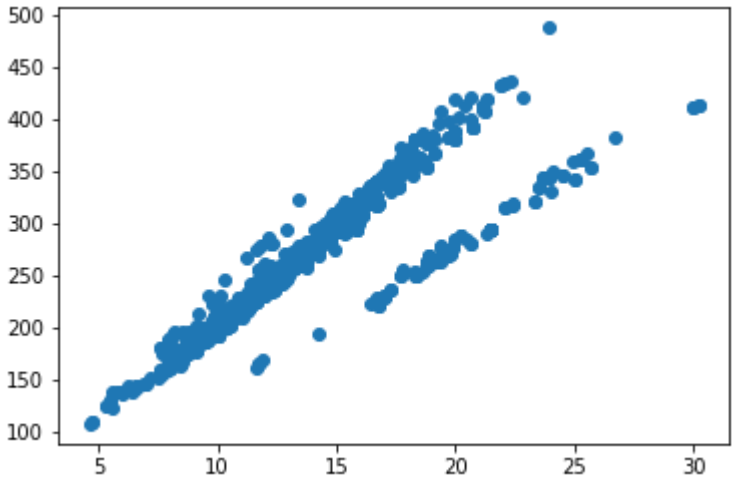


In [20]:

```
plt.scatter(x[:,5],y)
```

Out[20]:

```
<matplotlib.collections.PathCollection at 0x28c8c9dc048>
```

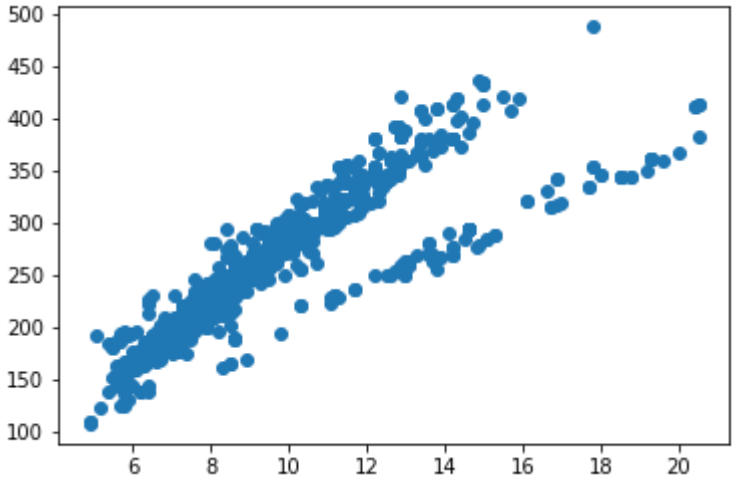


In [21]:

```
plt.scatter(x[:,6],y)
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x28c8ca44d48>
```

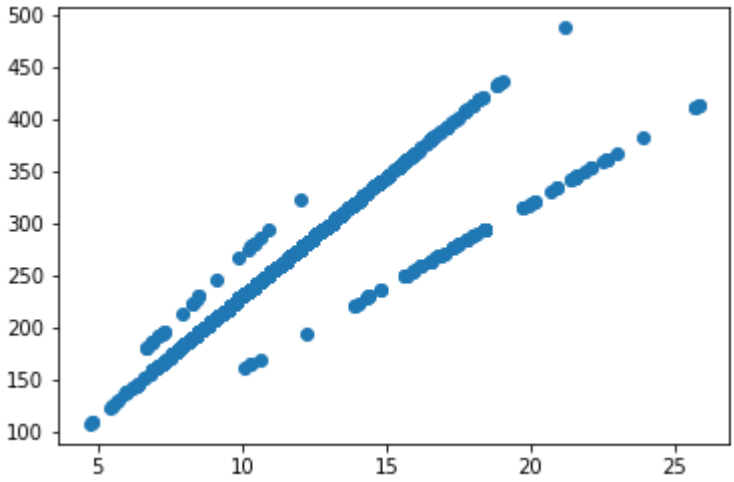


In [22]:

```
plt.scatter(x[:,7],y)
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x28c8caba848>
```



In [23]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [24]:

```
x_train
```

Out[24]:

```
array([[0.0, 1.0, 0.0, ..., 14.6, 10.2, 12.6],
       [0.0, 1.0, 0.0, ..., 16.9, 12.8, 15.1],
       [0.0, 0.0, 1.0, ..., 12.1, 8.7, 10.6],
       ...,
       [0.0, 0.0, 1.0, ..., 13.5, 9.6, 11.7],
       [0.0, 0.0, 1.0, ..., 15.8, 10.2, 13.3],
       [0.0, 1.0, 0.0, ..., 13.1, 9.3, 11.4]], dtype=object)
```

In [25]:

```
y_train
```

Out[25]:

```
array([290, 347, 244, 218, 170, 267, 389, 138, 218, 198, 280, 196, 237,
       255, 242, 168, 218, 246, 250, 186, 172, 315, 239, 413, 214, 260,
       246, 246, 264, 264, 163, 317, 254, 292, 380, 299, 230, 186, 216,
       232, 244, 202, 271, 170, 225, 162, 301, 225, 264, 214, 346, 212,
       191, 244, 242, 196, 207, 212, 288, 382, 294, 267, 235, 235, 322,
       196, 230, 259, 159, 224, 216, 235, 297, 181, 382, 239, 377, 294,
       223, 212, 435, 228, 230, 209, 294, 232, 294, 253, 239, 216, 168,
       222, 193, 283, 186, 340, 207, 193, 251, 209, 327, 271, 354, 258,
       267, 297, 271, 230, 251, 366, 200, 262, 168, 315, 212, 294, 207,
       207, 258, 196, 209, 221, 362, 389, 347, 308, 191, 198, 363, 260,
       235, 255, 294, 225, 232, 297, 232, 290, 251, 304, 182, 175, 277,
       294, 274, 260, 225, 163, 290, 230, 145, 205, 297, 196, 308, 168,
       380, 407, 338, 191, 264, 292, 225, 193, 336, 244, 317, 248, 359,
       196, 380, 179, 209, 237, 306, 235, 223, 338, 163, 317, 297, 237,
       184, 258, 338, 359, 235, 182, 260, 354, 244, 281, 324, 264, 165,
       419, 235, 382, 393, 216, 250, 258, 198, 421, 209, 354, 200, 271,
       209, 156, 259, 308, 258, 396, 317, 290, 276, 260, 290, 191, 274,
       317, 184, 363, 195, 202, 246, 315, 360, 184, 179, 340, 308, 196,
       221, 322, 354, 283, 221, 191, 214, 159, 254, 301, 182, 285, 276,
       361, 193, 251, 308, 179, 324, 352, 242, 246, 209, 336, 225, 216,
       200, 244, 317, 306, 184, 318, 209, 320, 246, 366, 255, 248, 269,
       258, 264, 346, 271, 350, 255, 189, 230, 161, 292, 218, 184, 253,
       288, 317, 347, 270, 237, 253, 228, 432, 294, 260, 244, 212, 228,
       170, 340, 225, 126, 213, 250, 294, 182, 278, 344, 414, 258, 315,
       297, 248, 189, 277, 232, 221, 207, 292, 294, 409, 264, 299, 230,
       402, 294, 362, 189, 216, 239, 276, 352, 177, 145, 283, 262, 165,
       175, 177, 209, 271, 380, 370, 382, 218, 267, 192, 198, 281, 228,
       285, 265, 380, 237, 186, 214, 196, 285, 175, 354, 320, 230, 281,
       281, 207, 414, 264, 246, 278, 377, 380, 209, 329, 278, 269, 269,
       320, 274, 274, 251, 290, 244, 200, 196, 255, 324, 281, 267, 262,
       354, 216, 398, 411, 251, 292, 228, 179, 292, 193, 264, 317, 269,
       292, 138, 230, 271, 356, 205, 283, 225, 209, 159, 212, 304, 278,
       235, 255, 177, 262, 198, 260, 393, 166, 310, 248, 228, 310, 253,
       373, 189, 198, 145, 239, 285, 205, 248, 338, 272, 198, 147, 198,
       207, 161, 331, 200, 320, 304, 216, 242, 262, 274, 179, 267, 235,
       216, 175, 198, 411, 179, 380, 297, 301, 179, 138, 380, 264, 278,
       230, 368, 235, 244, 246, 209, 297, 175, 269, 214, 347, 232, 338,
       177, 248, 267, 108, 246, 251, 294, 258, 242, 338, 235, 264, 359,
       223, 292, 340, 264, 232, 281, 380, 225, 207, 276, 294, 202, 283,
       301, 232, 292, 338, 251, 196, 327, 271, 285, 294, 244, 196, 184,
       184, 209, 297, 209, 260, 200, 177, 221, 216, 223, 223, 191, 179,
       301, 191, 168, 306, 221, 356, 196, 267, 200, 380, 216, 264, 222,
       310, 224, 214, 258, 283, 267, 340, 294, 386, 264, 225, 288, 193,
       253, 269, 200, 362, 290, 182, 196, 317, 285, 276, 368, 278, 200,
       278, 258, 228, 336, 193, 281, 207, 202, 228, 184, 152, 184, 253,
       347, 258, 380, 175, 237, 225, 244, 228, 384, 285, 239, 380, 175,
       228, 260, 253, 290, 225, 244, 189, 253, 334, 177, 258, 286, 221,
       239, 166, 177, 242, 329, 186, 184, 292, 237, 214, 336, 344, 207,
       212, 253, 488, 292, 248, 306, 230, 344, 306, 237, 393, 225, 242,
       248, 283, 196, 237, 317, 271, 317, 264, 255, 413, 380, 255, 380,
       292, 276, 235, 292, 228, 189, 172, 198, 301, 237, 277, 308, 278,
       262, 159, 315, 393, 200, 331, 297, 173, 419, 186, 260, 193, 202,
       255, 230, 251, 283, 232, 419, 221, 143, 275, 334, 230, 251, 225,
       221, 344, 294, 366, 179, 281, 232, 191, 197, 354, 227, 181, 251,
       269, 239, 253, 294, 221, 361, 230, 232, 285, 246, 179, 207, 306,
```

```
       272, 189, 194, 216, 317, 179, 143, 253, 202, 283, 255, 283, 230,
       179, 285, 294, 260, 255, 308, 192, 274, 267, 306, 221, 343, 152,
       189, 218, 269, 347, 322, 182, 225, 179, 230, 223, 218, 253, 235,
       251, 280, 202, 308, 340, 221, 179, 288, 435, 175, 212, 198, 322,
       246, 177, 320, 216, 409, 177, 207, 278, 163, 209, 147, 421, 297,
       315, 193, 278, 315, 193, 200, 232, 361, 317, 253, 276, 262, 182,
       274, 168, 191, 225, 189, 362, 304, 175, 320, 166, 207, 324, 161,
       242, 251, 308, 207, 197, 230, 191, 258, 274, 237, 255, 246, 299,
       225, 196, 294, 225, 196, 271, 313, 253, 196, 225, 209, 214, 432,
       407, 156, 212, 380, 322, 186, 267, 271, 340, 230, 281, 170, 186,
       251, 308, 242, 186, 256, 269, 306, 262], dtype=int64)
```

In [26]:

```
x_test
```

Out[26]:

```
array([[0.0, 0.0, 1.0, ..., 17.2, 13.5, 15.5],
       [0.0, 1.0, 0.0, ..., 10.5, 7.3, 9.1],
       [0.0, 1.0, 0.0, ..., 11.4, 8.3, 10.0],
       ...,
       [0.0, 0.0, 1.0, ..., 9.3, 7.0, 8.3],
       [0.0, 0.0, 1.0, ..., 9.1, 6.7, 8.0],
       [0.0, 1.0, 0.0, ..., 9.1, 6.7, 8.0]], dtype=object)
```

In [27]:

```
y_test
```

Out[27]:

```
array([356, 209, 230, 212, 168, 292, 212, 276, 202, 334, 313, 437, 224,
       281, 177, 260, 414, 223, 251, 359, 191, 189, 244, 242, 131, 283,
       274, 294, 246, 110, 359, 239, 229, 237, 191, 196, 294, 221, 237,
       237, 184, 184, 202, 194, 297, 198, 260, 179, 344, 359, 338, 288,
       290, 129, 230, 179, 283, 159, 258, 209, 207, 205, 225, 294, 262,
       299, 354, 230, 207, 124, 304, 189, 354, 270, 338, 216, 283, 179,
       235, 166, 186, 253, 161, 334, 407, 246, 191, 172, 290, 258, 262,
       209, 283, 342, 356, 368, 168, 221, 368, 262, 182, 320, 126, 166,
       202, 196, 200, 288, 191, 259, 214, 228, 269, 317, 327, 294, 292,
       244, 361, 200, 258, 191, 382, 147, 179, 310, 229, 237, 292, 264,
       230, 294, 184, 136, 344, 373, 283, 110, 198, 232, 262, 191, 342,
       368, 189, 175, 285, 345, 301, 138, 179, 251, 161, 235, 310, 242,
       292, 221, 324, 235, 177, 209, 177, 202, 242, 205, 264, 184, 400,
       334, 260, 285, 235, 301, 281, 276, 265, 242, 184, 345, 292, 184,
       317, 200, 347, 290, 205, 350, 184, 290, 267, 271, 239, 232, 218,
       269, 228, 216, 255, 317, 260, 225, 223, 329, 290, 288, 294, 258,
       380, 198, 221, 191, 184, 184], dtype=int64)
```

In [28]:

```
from sklearn.preprocessing import StandardScaler
sc1=StandardScaler()
x_train=sc1.fit_transform(x_train)
x_test=sc1.transform(x_test)
```

In [29]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [30]:

```
rf=RandomForestRegressor(n_estimators=10,criterion="mse",random_state=0,max_depth=5)
```

In [31]:

```
rf.fit(x_train,y_train)
```

Out[31]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=5, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=10, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

In [32]:

```
from joblib import dump
dump(rf,'RFR.save')
```

Out[32]:

```
['RFR.save']
```

In [33]:

```
y_pred=rf.predict(x_test)
y_pred
```

Out[33]:

```
array([354.41811702, 211.96427987, 234.63331749, 216.10662307,
       172.98801208, 292.94190629, 213.45233014, 281.39590092,
       199.02261235, 351.74868073, 312.94747388, 405.38876607,
       209.23012886, 279.37005348, 178.19672371, 260.15349384,
       399.76558633, 219.52565365, 264.95613472, 354.41811702,
       194.36111718, 193.76274356, 246.05032187, 245.7455449 ,
       128.31666667, 284.31421324, 271.51436073, 296.45119406,
       246.05032187, 123.14      , 354.41811702, 242.40393986,
       231.7239881 , 235.54893173, 194.36111718, 198.15792232,
       294.33919184, 219.52565365, 236.66498524, 238.17367488,
       187.01783499, 182.3309241 , 199.02261235, 175.72804979,
       294.33919184, 199.02261235, 260.64901853, 180.06698583,
       348.85689988, 354.41811702, 351.74868073, 280.59223893,
       288.70266239, 128.31666667, 234.63331749, 180.06698583,
       282.79280745, 170.43183976, 258.87180669, 211.96427987,
       206.2959088 , 206.2959088 , 222.36729626, 296.45119406,
       260.64901853, 295.45808073, 356.83376845, 232.4239881 ,
       206.2959088 , 128.31666667, 306.22323319, 192.6823514 ,
       356.83376845, 292.77257564, 345.97938563, 216.63662121,
       284.31421324, 178.19672371, 235.54893173, 172.98801208,
       187.01783499, 255.54094284, 170.43183976, 344.08563563,
       378.51376652, 246.05032187, 194.36111718, 177.32694812,
       296.45119406, 263.5890871 , 260.64901853, 211.96427987,
       284.31421324, 348.85689988, 354.41811702, 354.41811702,
       172.98801208, 220.38858294, 354.41811702, 260.64901853,
       181.58991027, 318.09577257, 123.14      , 172.98801208,
       199.02261235, 198.15792232, 199.02261235, 286.00307542,
       194.36111718, 260.95075127, 215.4083345 , 233.09639441,
       271.12024308, 312.94747388, 321.86836661, 291.79263087,
       294.33919184, 246.40935564, 356.83376845, 198.15792232,
       258.31004198, 194.36111718, 388.84823259, 145.4781746 ,
       180.06698583, 306.78551777, 231.75732143, 236.25580628,
       290.29790048, 262.98762016, 234.63331749, 294.33919184,
       187.01783499, 141.13103175, 348.85689988, 371.43680167,
       284.31421324, 123.14      , 198.15792232, 234.63331749,
       252.26395267, 194.36111718, 348.85689988, 354.41811702,
       192.08655618, 178.19672371, 284.31421324, 352.65934509,
       300.86578267, 141.13103175, 180.06698583, 252.72128747,
       171.84612547, 236.66498524, 308.01145434, 247.32779711,
       290.29790048, 219.52565365, 321.86836661, 238.17367488,
       177.32694812, 211.96427987, 178.19672371, 205.67033437,
       244.83222232, 206.2959088 , 263.71082792, 183.65632977,
       395.7198874 , 348.85689988, 260.64901853, 284.31421324,
       235.54893173, 300.86578267, 280.47204905, 281.39590092,
       262.98762016, 244.83222232, 183.65632977, 355.07499652,
       294.33919184, 182.3309241 , 318.09577257, 198.15792232,
       324.31743697, 288.70266239, 206.2959088 , 356.83376845,
       182.3309241 , 294.33919184, 267.93528233, 271.51436073,
       241.28788635, 234.63331749, 218.62430998, 274.9402315 ,
       233.09639441, 216.63662121, 250.01908349, 318.09577257,
       260.64901853, 222.36729626, 220.38858294, 326.92621749,
       293.35476343, 290.65517647, 291.79263087, 258.31004198,
       383.22505285, 198.15792232, 219.52565365, 194.36111718,
       183.65632977, 182.3309241 ])
```

In [34]:

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[34]:

```
0.9913047727968988
```

In [ ]:

In [ ]: