```
#
```

simple regression problem

problem to pridict SAT with GPA

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt


import seaborn as sns
```

second step to read the data set from your directory

```
sat =pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/SAT%20GPA.csv')
```

step 3 to analysis the data you imported

```
sat.head()
```

✓ 0s completed at 4:15 PM

0	1270	3.4
1	1220	4.0
2	1160	3.8
3	950	3.8
4	1070	4.0

we analysis the data set that any value is missing or not any value is missing then drop these value and check the missing values to info function

```
sat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   SAT     1000 non-null   int64  
 1   GPA     1000 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 15.8 KB
```

```
sat.describe()
```

	SAT	GPA
count	1000.000000	1000.000000
mean	1033.290000	3.203700

	SAT	GPA
std	142.873681	0.542541
min	530.000000	1.800000
25%	930.000000	2.800000
50%	1030.000000	3.200000
75%	1130.000000	3.700000
max	1440.000000	4.500000

sat.corr()

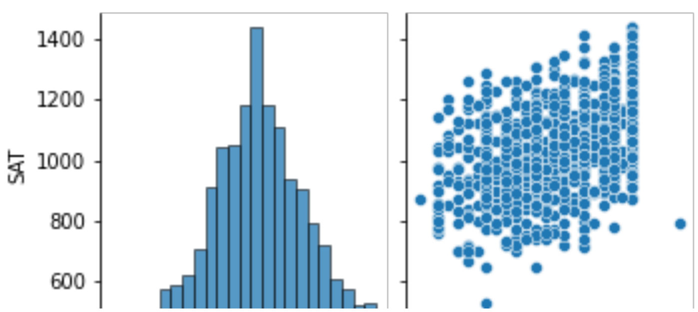
```
sat.corr()
```

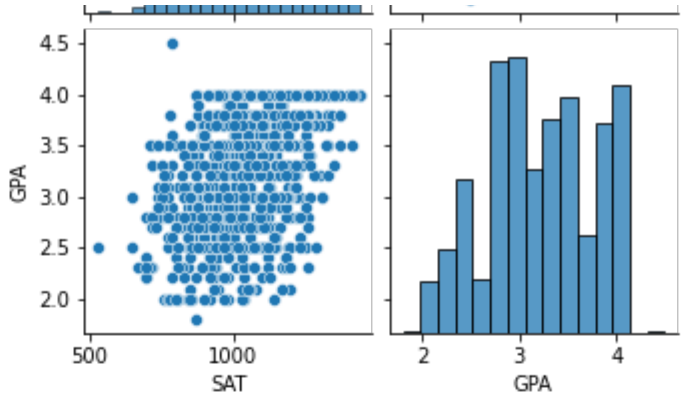
	SAT	GPA
SAT	1.000000	0.429649
GPA	0.429649	1.000000

step 4 to ploating the data set to use of seaborn library

```
sns.pairplot(sat)
```

```
<seaborn.axisgrid.PairGrid at 0x7f99de9b4490>
```





after visualization the data after lets define the y and x

```
sat.columns
```

```
Index(['SAT', 'GPA'], dtype='object')
```

```
y=sat['SAT']
```

```
y.shape
```

```
(1000,)
```

```
x=sat[['GPA']]
```

```
x.shape
```

```
(1000, 1)
```

after define x and y to train splite

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=2529)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((700, 1), (300, 1), (700,), (300,))
```

check the random

```
x_train
```

	GPA
669	3.7
583	3.7
688	2.8
422	3.9
825	4.0
...	...
740	2.5
399	2.6
828	3.2
562	2.7
352	3.0

700 rows × 1 columns

all these above step are same for all

after we are import the model

from sklearn.linear_model import LinearRegression

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

```
reg=LinearRegression()
```

after importing the regression model to fit our data set

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

```
LinearRegression()
```

after fit the model to pridict

```
reg.intercept_
```

```
673.2291896122774
```

```
reg.coef_
```

```
array([111.01584994])
```

```
reg.predict(x_test)
```

```
array([1061.78466441, 1095.08941939, 1050.68307942, 1117.29258938,
       1095.08941939, 1061.78466441, 1006.27673944, 1083.9878344 ,
        895.2608895 , 1095.08941939, 1017.37832443, 1117.29258938,
       1017.37832443,  961.87039946,  972.97198446, 1095.08941939,
       1039.58149442,  950.76881447, 1095.08941939, 1006.27673944,
       1039.58149442, 1006.27673944,  984.07356945,  972.97198446,
       1095.08941939,  995.17515445,  928.56564448,  972.97198446,
       1083.9878344 , 1061.78466441, 1106.19100439, 1083.9878344 ,
        984.07356945,  972.97198446,  972.97198446,  995.17515445,
       1095.08941939, 1117.29258938,  950.76881447, 1017.37832443,
       1061.78466441,  984.07356945,  972.97198446, 1006.27673944,
       1050.68307942, 1017.37832443, 1083.9878344 , 1117.29258938,
        972.97198446, 1117.29258938,  972.97198446, 1061.78466441,
        984.07356945, 1006.27673944, 1117.29258938, 1117.29258938,
       1117.29258938, 1050.68307942,  950.76881447, 1117.29258938,
        950.76881447, 1117.29258938, 1083.9878344 ,  984.07356945,
       1028.47990943, 1039.58149442, 1095.08941939,  984.07356945,
       1050.68307942,  984.07356945, 1039.58149442,  950.76881447,
       1039.58149442,  950.76881447, 1117.29258938, 1017.37832443,
       1050.68307942, 1117.29258938, 1117.29258938, 1006.27673944,
       1006.27673944, 1117.29258938,  972.97198446, 1017.37832443,
        984.07356945, 1117.29258938,  972.97198446, 1072.88624941,
       1050.68307942,  917.46405949, 1006.27673944, 1095.08941939,
       1095.08941939, 1028.47990943, 1039.58149442,  950.76881447,
       1028.47990943,  995.17515445, 1117.29258938, 1028.47990943,
        984.07356945, 1061.78466441,  950.76881447,  984.07356945,
        928.56564448, 1061.78466441,  972.97198446,  984.07356945,
        972.97198446, 1028.47990943, 1028.47990943, 1072.88624941,
       1061.78466441, 1006.27673944, 1061.78466441, 1117.29258938,
       1117.29258938, 1061.78466441,  961.87039946, 1061.78466441,
       1006.27673944,  995.17515445, 1095.08941939,  984.07356945,
        906.36247449, 1083.9878344 , 1061.78466441,  895.2608895 ,
       1039.58149442,  961.87039946, 1095.08941939, 1117.29258938,
       1006.27673944, 1083.9878344 ,  984.07356945,  950.76881447
```

```
       1006.27673944, 1083.9878344 ,  984.07356945,  950.76881447,
       1061.78466441, 1061.78466441, 1095.08941939, 1095.08941939,
       1117.29258938, 1117.29258938, 1006.27673944, 1117.29258938,
        995.17515445, 1028.47990943, 1006.27673944,  984.07356945,
       1095.08941939,  961.87039946, 1117.29258938, 1028.47990943,
       1061.78466441, 1028.47990943, 1061.78466441,  961.87039946,
       1006.27673944, 1006.27673944, 1017.37832443, 1095.08941939,
        950.76881447, 1039.58149442,  984.07356945,  950.76881447,
       1006.27673944,  895.2608895 ,  984.07356945, 1095.08941939,
       1095.08941939,  939.66722947,  950.76881447,  984.07356945,
       1095.08941939,  895.2608895 ,  961.87039946,  950.76881447,
        984.07356945,  917.46405949, 1006.27673944,  928.56564448,
       1117.29258938,  984.07356945, 1117.29258938, 1006.27673944,
        939.66722947, 1072.88624941,  984.07356945, 1006.27673944,
       1061.78466441, 1006.27673944, 1061.78466441,  950.76881447,
       1117.29258938, 1039.58149442,  984.07356945, 1006.27673944,
       1050.68307942, 1072.88624941, 1006.27673944,  917.46405949,
       1083.9878344 , 1061.78466441,  928.56564448, 1039.58149442,
       1061.78466441, 1117.29258938,  984.07356945, 1017.37832443,
       1039.58149442, 1061.78466441, 1039.58149442, 1028.47990943,
        950.76881447,  972.97198446, 1117.29258938,  972.97198446,
       1095.08941939, 1039.58149442, 1095.08941939, 1083.9878344 ,
        972.97198446, 1006.27673944,  928.56564448, 1039.58149442,
        995.17515445,  939.66722947, 1072.88624941,  928.56564448,
       1061.78466441, 1028.47990943, 1017.37832443,  895.2608895 ,
```

```
y_prid=reg.predict(x_test)
```

```
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error
```

```
mean_absolute_error(y_test,y_prid)
```

```
105.93877473699905
```

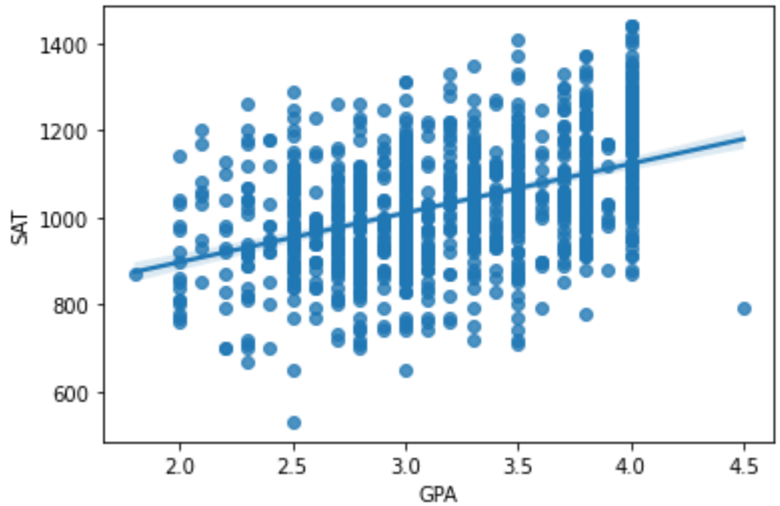
```
mean_absolute_percentage_error(y_test,y_prid)
```

```
0.10467104034918914
```

and visulation the data through sns library

```
sns.regplot(x='GPA',y='SAT', data=sat)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f99d6b84650>
```



multipal regression

```
import pandas as pd
```

```
import numpy as np
```

```
import sklearn as sns1
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

read the data set

```
df=pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Boston.csv')
```

```
df.head()
```

	CRIM	ZN	INDUS	CHAS	NX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64
 4   NX       506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
```

```
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NX	RM	AGE	DIS	RAD	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.23
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.53
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.00
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.00
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.00
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.00
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.00

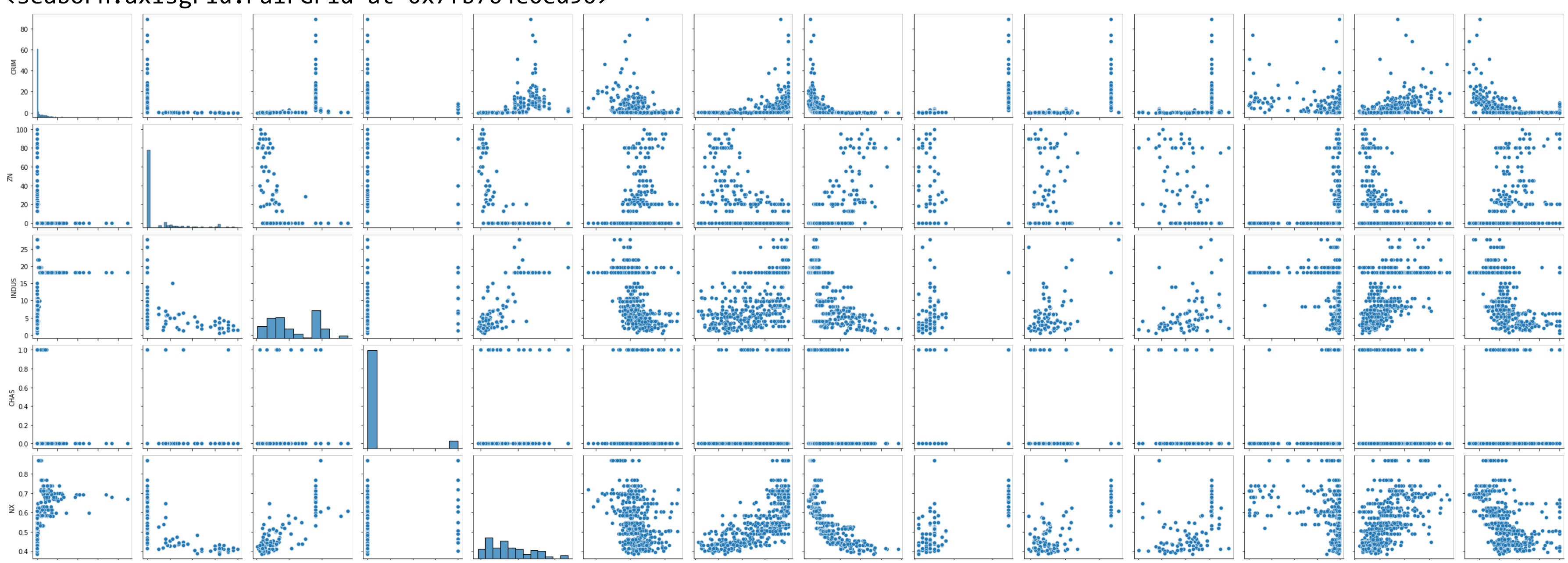
```
df.corr()
```

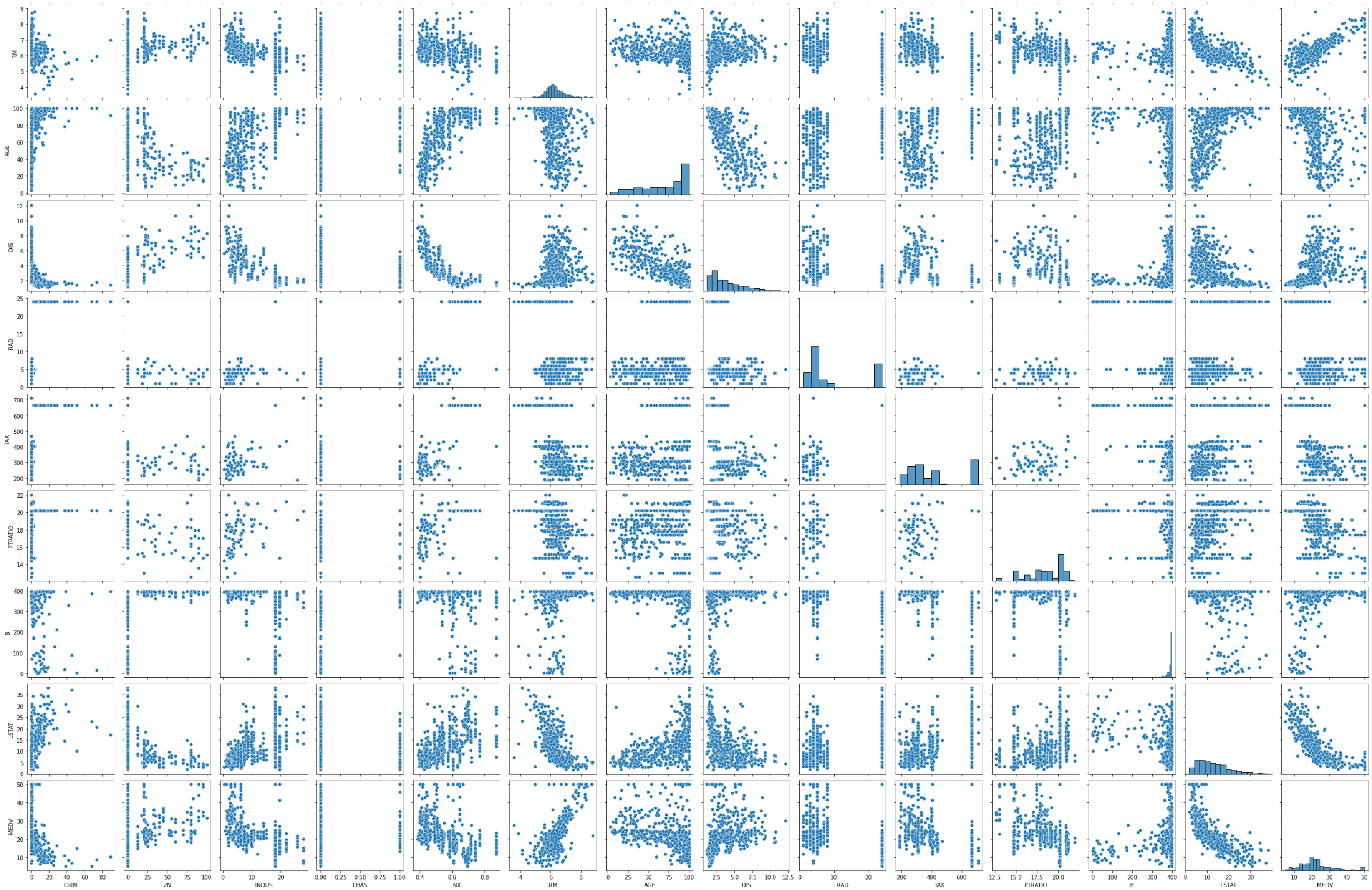
	CRIM	ZN	INDUS	CHAS	NX	RM	AGE	DIS	RAD	TAX	PTRA
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121

NX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374
MEDV	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fb764e0ed90>
```




RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
MEDV
CRIM
ZN
INDUS
CHAS
NX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
MEDV

```
df.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

```
y=df['MEDV']
```

```
x=df[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT']]
```

```
x.shape
```

```
(506, 13)
```

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=2529)
```

after train the data we are standerlization

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
```

```
x_train=sc.fit_transform(x_train)
```

```
x_test=sc.fit_transform(x_test)
```

```
x_train
```

```
array([[-0.14113619, -0.48175769, -0.19860022, ...,  0.00438903,
        -0.05084503, -0.01555641],
       [-0.42121529,  3.02166196, -1.33410259, ..., -1.68641979,
         0.42969249, -1.33650784],
       [-0.41266839, -0.48175769,  0.22414717, ...,  0.14148164,
         0.19739169, -0.10842497],
       ...,
       [-0.38944304, -0.48175769, -0.19860022, ...,  0.00438903,
         0.37963873,  0.77313338],
       [-0.41404001,  0.41002186, -0.81324318, ..., -0.72677154,
         0.43161763,  0.09671754],
       [-0.41578561,  2.06618387, -1.3831586 , ..., -0.04130851,
         0.39707198, -0.68781395]])
```

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

```
model=LinearRegression()
```

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

```
LinearRegression()
```

```
model.intercept_
```

```
22.83248587570622
```

```
model.coef_
```

```
array([-1.20767891,  0.85995285,  0.1070255 ,  0.63555228, -2.43159195,
        3.08829222,  0.13082323, -3.31025945,  2.22711291, -1.65403572,
       -2.10989321,  0.94408913, -3.91890566])
```

```
df.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO'. 'B'. 'LSTAT'. 'MEDV'].
```

```
       dtype='object')
```

```
y_prid=model.predict(x_test)
```

```
from sklearn.metrics import mean_absolute_percentage_error,r2_score
```

```
mean_absolute_percentage_error(y_test,y_prid)
```

```
0.18900464575924525
```

```
r2_score(y_test,y_prid)
```

```
0.5945114562128394
```