In [64]:
```
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
```

In [65]:
```
housing=pd.read_csv("data.csv")
housing
```

Out[65]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.1
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.0
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.9
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.3
...	...	...	...	...	...	...	...	...	...	...	...	...	.
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.6
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.0
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.6
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.4
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.8

506 rows × 14 columns

Out[65]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.1
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.0
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.9
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.3
...	...	...	...	...	...	...	...	...	...	...	...	...	.
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.6
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.0
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.6
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.4
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.8

506 rows × 14 columns

In [66]:
```
housing.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   NOX      506 non-null    float64
 5   RM       501 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 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(11), int64(3)
memory usage: 55.5 KB
<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   NOX      506 non-null    float64
 5   RM       501 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 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(11), int64(3)
memory usage: 55.5 KB
```

In [67]:
```
housing.isnull().sum()
```

Out[67]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         5
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

Out[67]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         5
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [68]:
```
housing.describe()
```

Out[68]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	501.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284341	68.574901	3.79
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.705587	28.148861	2.10
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.884000	45.025000	2.10
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208000	77.500000	3.20
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.625000	94.075000	5.18
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

Out[68]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	501.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284341	68.574901	3.79
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.705587	28.148861	2.10
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.884000	45.025000	2.10
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208000	77.500000	3.20
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.625000	94.075000	5.18
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

In [69]:
```
housing["CHAS"].value_counts()
```

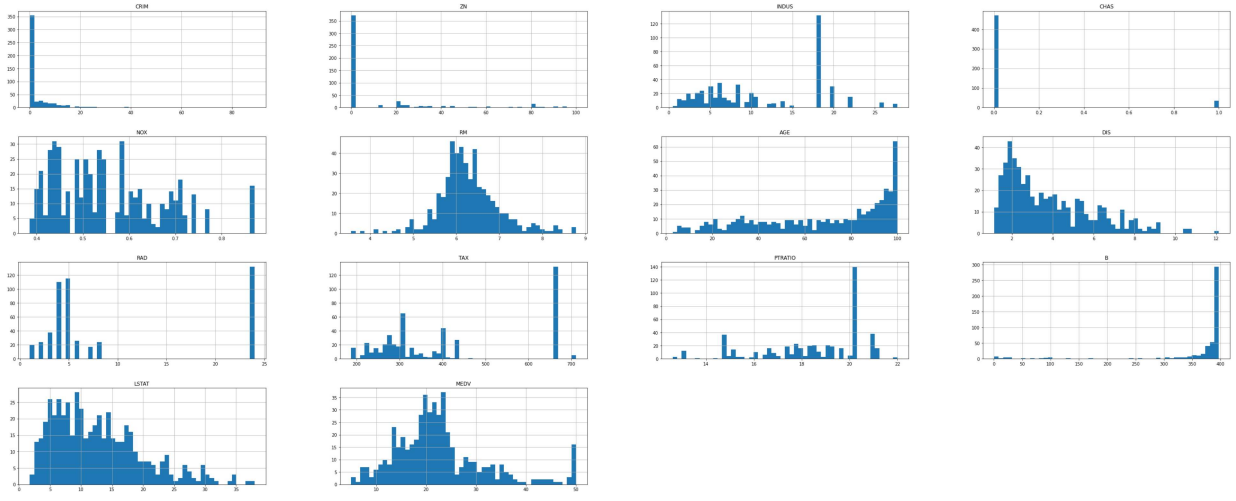
Out[69]:
```
0    471
1     35
Name: CHAS, dtype: int64
```

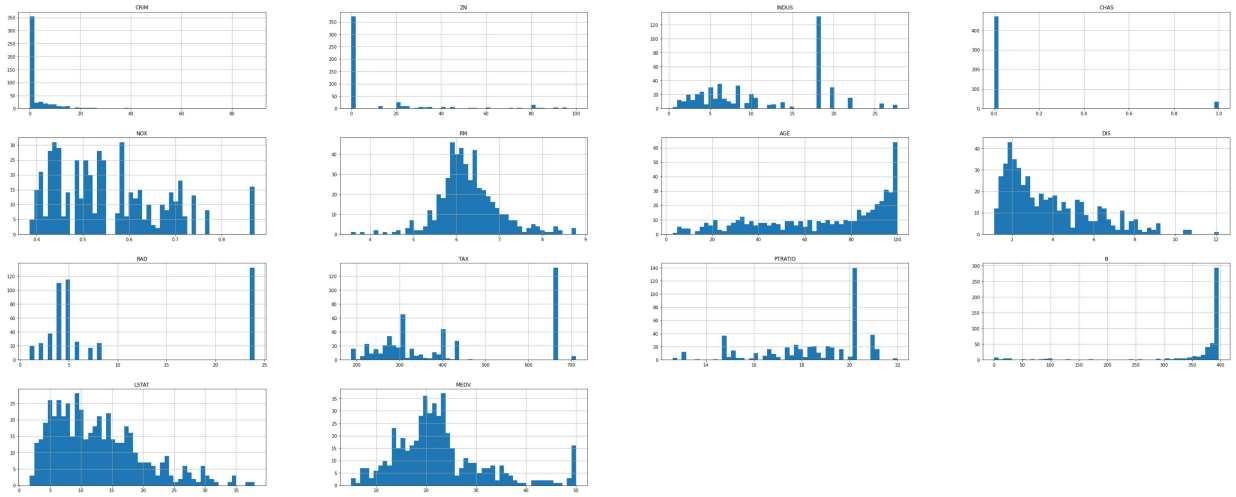
Out[69]:
```
0    471
1     35
Name: CHAS, dtype: int64
```

In [70]:
```
housing.hist(bins=50, figsize=( 50,20))
```

Out[70]:
```
array([[<AxesSubplot:title={'center':'CRIM'}>,
        <AxesSubplot:title={'center':'ZN'}>,
        <AxesSubplot:title={'center':'INDUS'}>,
        <AxesSubplot:title={'center':'CHAS'}>],
       [<AxesSubplot:title={'center':'NOX'}>,
        <AxesSubplot:title={'center':'RM'}>,
        <AxesSubplot:title={'center':'AGE'}>,
        <AxesSubplot:title={'center':'DIS'}>],
       [<AxesSubplot:title={'center':'RAD'}>,
        <AxesSubplot:title={'center':'TAX'}>,
        <AxesSubplot:title={'center':'PTRATIO'}>,
        <AxesSubplot:title={'center':'B'}>],
       [<AxesSubplot:title={'center':'LSTAT'}>,
        <AxesSubplot:title={'center':'MEDV'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```

Out[70]:
```
array([[<AxesSubplot:title={'center':'CRIM'}>,
        <AxesSubplot:title={'center':'ZN'}>,
        <AxesSubplot:title={'center':'INDUS'}>,
        <AxesSubplot:title={'center':'CHAS'}>],
       [<AxesSubplot:title={'center':'NOX'}>,
        <AxesSubplot:title={'center':'RM'}>,
        <AxesSubplot:title={'center':'AGE'}>,
        <AxesSubplot:title={'center':'DIS'}>],
       [<AxesSubplot:title={'center':'RAD'}>,
        <AxesSubplot:title={'center':'TAX'}>,
        <AxesSubplot:title={'center':'PTRATIO'}>,
        <AxesSubplot:title={'center':'B'}>],
       [<AxesSubplot:title={'center':'LSTAT'}>,
        <AxesSubplot:title={'center':'MEDV'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```





In [71]:
```
import numpy as np
def split_train_test(data, test_ratio):
    shuffled =np.random.permutation(len(data))
    test_set_size=int(len(data)*test_ratio)
    test_indices=shuffled[:test_set_size]
    train_indices=shuffled[test_set_size:]
    return data.iloc[train_indices],data.iloc[test_indices]
```

In [72]:
```
  train_set, test_set = split_train_test(housing,0.2)
```

In [73]:
```
print(f"Rows in train set: {len(train_set)}\n Rows in test set:{len(test_set)}\n'
```

```
Rows in train set: 405
 Rows in test set:101

Rows in train set: 405
 Rows in test set:101
```

In [74]:
```
from sklearn.model_selection import train_test_split
train_set,test_set=train_test_split(housing,test_size=0.2,random_state=42)
print(f"ROWS IN TRAIN SET :{len(train_set)}\nROWS IN TEST SET :{len(test_set)}\n
```

```
ROWS IN TRAIN SET :404
ROWS IN TEST SET :102

ROWS IN TRAIN SET :404
ROWS IN TEST SET :102
```

In [75]:
```
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)
for train_index, test_index in split.split(housing, housing['CHAS']):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
```

In [76]:
```
strat_train_set.describe()
```

Out[76]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	404.000000	404.000000	404.000000	404.000000	404.000000	399.000000	404.000000	404.00
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.279481	69.039851	3.74
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.716784	28.258248	2.09
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.12
25%	0.086962	0.000000	5.190000	0.000000	0.453000	5.876500	44.850000	2.03
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.209000	78.200000	3.12
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.630500	94.100000	5.10
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

Out[76]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	404.000000	404.000000	404.000000	404.000000	404.000000	399.000000	404.000000	404.00
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.279481	69.039851	3.74
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.716784	28.258248	2.09
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.12
25%	0.086962	0.000000	5.190000	0.000000	0.453000	5.876500	44.850000	2.03
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.209000	78.200000	3.12
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.630500	94.100000	5.10
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

In [77]:
```
strat_test_set.describe()
```

Out[77]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000	102.00
mean	3.655942	13.450980	10.312255	0.068627	0.541353	6.303353	66.733333	3.98
std	10.400966	27.503241	6.761154	0.254068	0.111397	0.662996	27.772183	2.13
min	0.009060	0.000000	0.460000	0.000000	0.385000	4.138000	6.500000	1.13
25%	0.057827	0.000000	4.950000	0.000000	0.448000	5.912750	45.850000	2.22
50%	0.176150	0.000000	7.760000	0.000000	0.515000	6.176000	71.100000	3.42
75%	2.061955	0.000000	18.100000	0.000000	0.612750	6.539500	93.500000	5.60
max	88.976200	90.000000	27.740000	1.000000	0.871000	8.725000	100.000000	10.58

Out[77]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000	102.00
mean	3.655942	13.450980	10.312255	0.068627	0.541353	6.303353	66.733333	3.98
std	10.400966	27.503241	6.761154	0.254068	0.111397	0.662996	27.772183	2.13
min	0.009060	0.000000	0.460000	0.000000	0.385000	4.138000	6.500000	1.13
25%	0.057827	0.000000	4.950000	0.000000	0.448000	5.912750	45.850000	2.22
50%	0.176150	0.000000	7.760000	0.000000	0.515000	6.176000	71.100000	3.42
75%	2.061955	0.000000	18.100000	0.000000	0.612750	6.539500	93.500000	5.60
max	88.976200	90.000000	27.740000	1.000000	0.871000	8.725000	100.000000	10.58

In [78]:
```
strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 404 entries, 254 to 216
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     404 non-null    float64
 1   ZN       404 non-null    float64
 2   INDUS    404 non-null    float64
 3   CHAS     404 non-null    int64  
 4   NOX      404 non-null    float64
 5   RM       399 non-null    float64
 6   AGE      404 non-null    float64
 7   DIS      404 non-null    float64
 8   RAD      404 non-null    int64  
 9   TAX      404 non-null    int64  
 10  PTRATIO  404 non-null    float64
 11  B        404 non-null    float64
 12  LSTAT    404 non-null    float64
 13  MEDV     404 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 47.3 KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 404 entries, 254 to 216
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     404 non-null    float64
 1   ZN       404 non-null    float64
 2   INDUS    404 non-null    float64
 3   CHAS     404 non-null    int64  
 4   NOX      404 non-null    float64
 5   RM       399 non-null    float64
 6   AGE      404 non-null    float64
 7   DIS      404 non-null    float64
 8   RAD      404 non-null    int64  
 9   TAX      404 non-null    int64  
 10  PTRATIO  404 non-null    float64
 11  B        404 non-null    float64
 12  LSTAT    404 non-null    float64
 13  MEDV     404 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 47.3 KB
```

In [79]:
```
strat_test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 102 entries, 342 to 218
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     102 non-null    float64
 1   ZN       102 non-null    float64
 2   INDUS    102 non-null    float64
 3   CHAS     102 non-null    int64  
 4   NOX      102 non-null    float64
 5   RM       102 non-null    float64
 6   AGE      102 non-null    float64
 7   DIS      102 non-null    float64
 8   RAD      102 non-null    int64  
 9   TAX      102 non-null    int64  
 10  PTRATIO  102 non-null    float64
 11  B        102 non-null    float64
 12  LSTAT    102 non-null    float64
 13  MEDV     102 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 12.0 KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 102 entries, 342 to 218
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     102 non-null    float64
 1   ZN       102 non-null    float64
 2   INDUS    102 non-null    float64
 3   CHAS     102 non-null    int64  
 4   NOX      102 non-null    float64
 5   RM       102 non-null    float64
 6   AGE      102 non-null    float64
 7   DIS      102 non-null    float64
 8   RAD      102 non-null    int64  
 9   TAX      102 non-null    int64  
 10  PTRATIO  102 non-null    float64
 11  B        102 non-null    float64
 12  LSTAT    102 non-null    float64
 13  MEDV     102 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 12.0 KB
```

In [80]:
```
strat_train_set['CHAS'].value_counts()
```

Out[80]:
```
0    376
1     28
Name: CHAS, dtype: int64
```

Out[80]:
```
0    376
1     28
Name: CHAS, dtype: int64
```

In [81]:
```
strat_test_set['CHAS'].value_counts()
```

Out[81]:
```
0    95
1     7
Name: CHAS, dtype: int64
```

Out[81]:
```
0    95
1     7
Name: CHAS, dtype: int64
```

In [82]:
```
housing=strat_train_set.copy()
```

LOOKING FOR CORELATIONS

In [83]:
```
corr_matrix=housing.corr()
```

In [84]:
```
corr_matrix['MEDV'].sort_values(ascending=False)
```

Out[84]:
```
MEDV       1.000000
RM         0.680857
B          0.361761
ZN         0.339741
DIS        0.240451
CHAS       0.205066
AGE       -0.364596
RAD       -0.374693
CRIM      -0.393715
NOX       -0.422873
TAX       -0.456657
INDUS     -0.473516
PTRATIO   -0.493534
LSTAT     -0.740494
Name: MEDV, dtype: float64
```

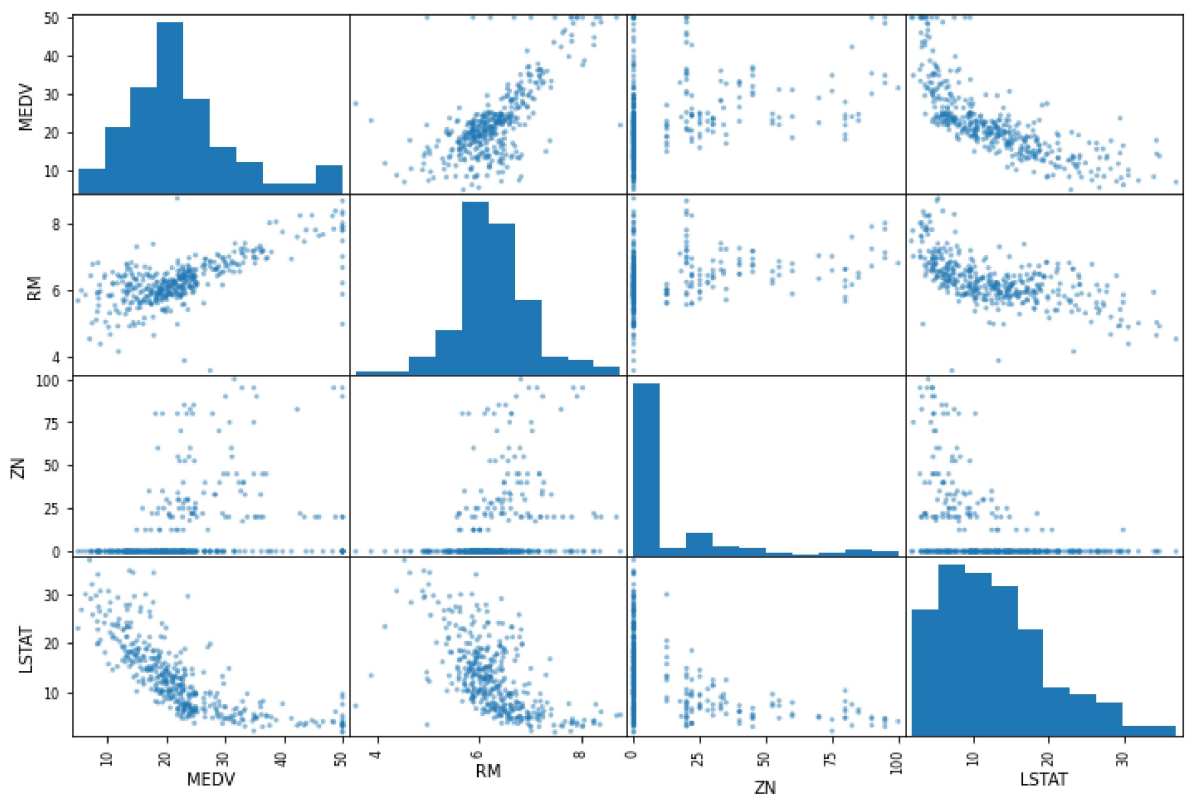
Out[84]:
```
MEDV       1.000000
RM         0.680857
B          0.361761
ZN         0.339741
DIS        0.240451
CHAS       0.205066
AGE       -0.364596
RAD       -0.374693
CRIM      -0.393715
NOX       -0.422873
TAX       -0.456657
INDUS     -0.473516
PTRATIO   -0.493534
LSTAT     -0.740494
Name: MEDV, dtype: float64
```

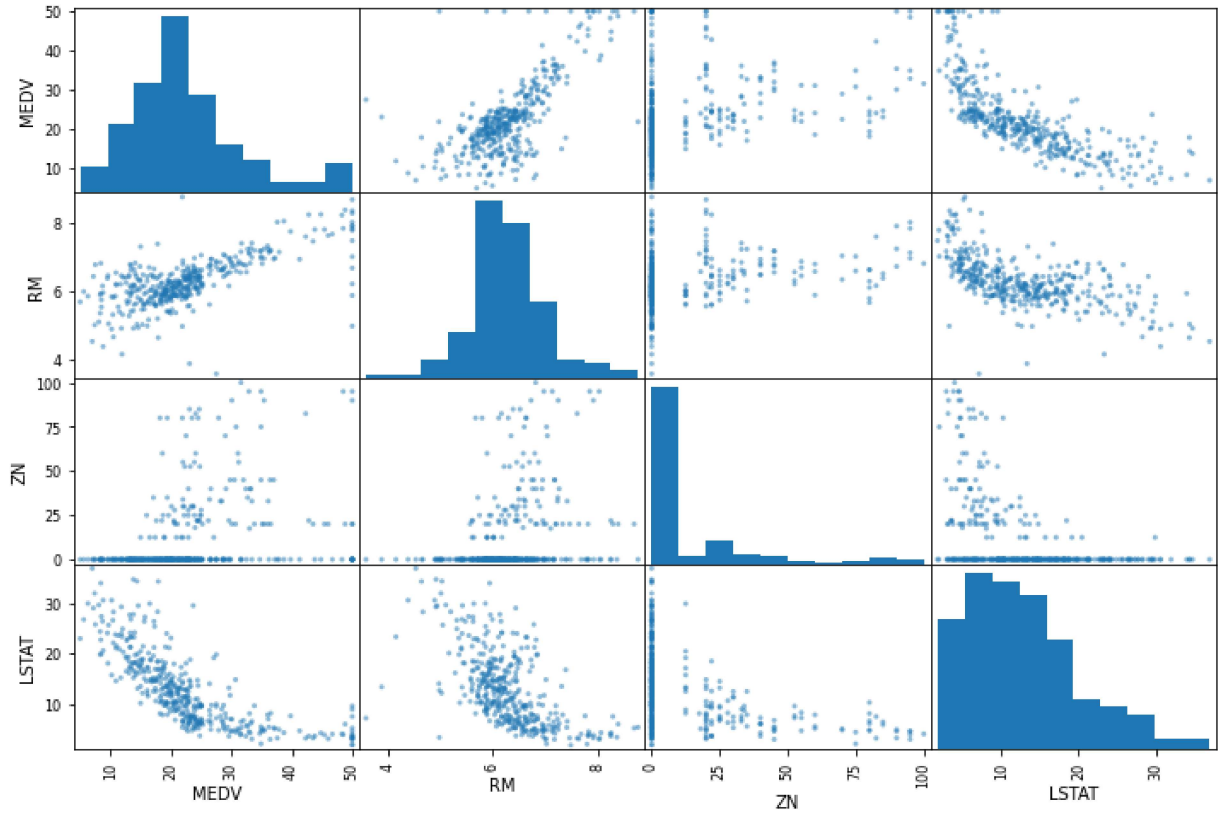
In [85]:
```
from pandas.plotting import scatter_matrix
```

In [86]:
```
attributes=["MEDV","RM","ZN","LSTAT"]
scatter_matrix(housing[attributes],figsize=(12,8))
```

Out[86]:
```
array([[<AxesSubplot:xlabel='MEDV', ylabel='MEDV'>,
        <AxesSubplot:xlabel='RM', ylabel='MEDV'>,
        <AxesSubplot:xlabel='ZN', ylabel='MEDV'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='MEDV'>],
       [<AxesSubplot:xlabel='MEDV', ylabel='RM'>,
        <AxesSubplot:xlabel='RM', ylabel='RM'>,
        <AxesSubplot:xlabel='ZN', ylabel='RM'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='RM'>],
       [<AxesSubplot:xlabel='MEDV', ylabel='ZN'>,
        <AxesSubplot:xlabel='RM', ylabel='ZN'>,
        <AxesSubplot:xlabel='ZN', ylabel='ZN'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='ZN'>],
       [<AxesSubplot:xlabel='MEDV', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='RM', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='ZN', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='LSTAT'>]], dtype=object)
```

Out[86]:
```
array([[<AxesSubplot:xlabel='MEDV', ylabel='MEDV'>,
        <AxesSubplot:xlabel='RM', ylabel='MEDV'>,
        <AxesSubplot:xlabel='ZN', ylabel='MEDV'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='MEDV'>],
       [<AxesSubplot:xlabel='MEDV', ylabel='RM'>,
        <AxesSubplot:xlabel='RM', ylabel='RM'>,
        <AxesSubplot:xlabel='ZN', ylabel='RM'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='RM'>],
       [<AxesSubplot:xlabel='MEDV', ylabel='ZN'>,
        <AxesSubplot:xlabel='RM', ylabel='ZN'>,
        <AxesSubplot:xlabel='ZN', ylabel='ZN'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='ZN'>],
       [<AxesSubplot:xlabel='MEDV', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='RM', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='ZN', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='LSTAT'>]], dtype=object)
```

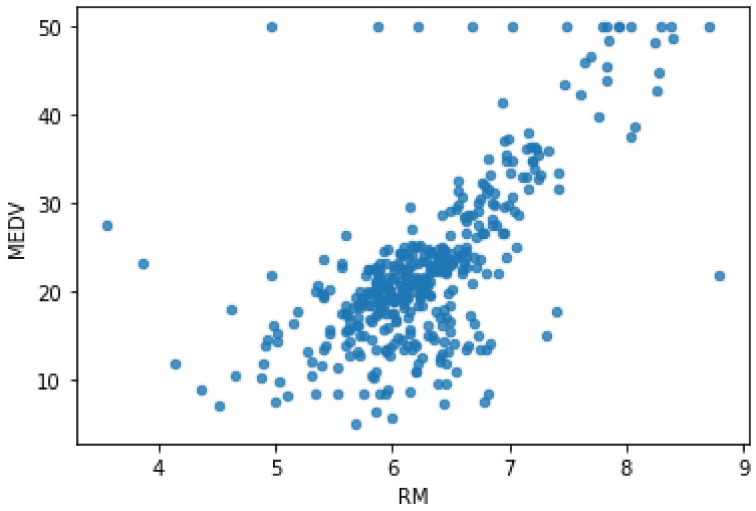

MEDV
RM
ZN
LSTAT

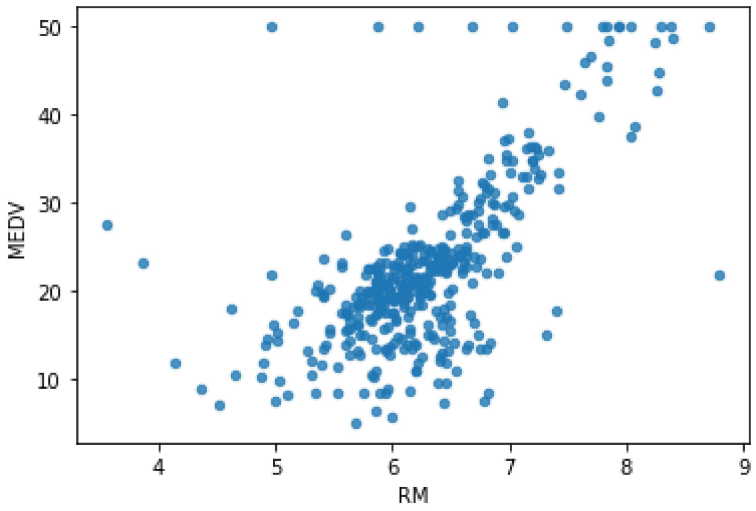

MEDV
RM
ZN
LSTAT

In [87]:
```
housing.plot(kind="scatter",x="RM",y="MEDV",alpha=0.8)
```

Out[87]: <AxesSubplot:xlabel='RM', ylabel='MEDV'>

Out[87]: <AxesSubplot:xlabel='RM', ylabel='MEDV'>





In [88]:
```
housing["TAXRM"] = housing["TAX"]/housing["RM"]
```

In [89]:
```
housing["TAXRM"]
```

Out[89]:
```
254     51.571709
348     42.200452
476    102.714374
321     45.012547
326     45.468948
          ...
155     65.507152
423    109.126659
98      35.294118
455    102.068966
216     46.875000
Name: TAXRM, Length: 404, dtype: float64
```

Out[89]:
```
254     51.571709
348     42.200452
476    102.714374
321     45.012547
326     45.468948
          ...
155     65.507152
423    109.126659
98      35.294118
455    102.068966
216     46.875000
Name: TAXRM, Length: 404, dtype: float64
```

In [90]:
```
housing.head()
```

Out[90]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
254	0.04819	80.0	3.64	0	0.392	6.108	32.0	9.2203	1	315	16.4	392.89	6.5
348	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	390.94	5.9
476	4.87141	0.0	18.10	0	0.614	6.484	93.6	2.3053	24	666	20.2	396.21	18.6
321	0.18159	0.0	7.38	0	0.493	6.376	54.3	4.5404	5	287	19.6	396.90	6.8
326	0.30347	0.0	7.38	0	0.493	6.312	28.9	5.4159	5	287	19.6	396.90	6.1

Out[90]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
254	0.04819	80.0	3.64	0	0.392	6.108	32.0	9.2203	1	315	16.4	392.89	6.5
348	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	390.94	5.9
476	4.87141	0.0	18.10	0	0.614	6.484	93.6	2.3053	24	666	20.2	396.21	18.6
321	0.18159	0.0	7.38	0	0.493	6.376	54.3	4.5404	5	287	19.6	396.90	6.8
326	0.30347	0.0	7.38	0	0.493	6.312	28.9	5.4159	5	287	19.6	396.90	6.1

LOOKING FOR COORELATION

In [91]:
```
corr_matrix=housing.corr()
corr_matrix['MEDV'].sort_values(ascending=False)
```

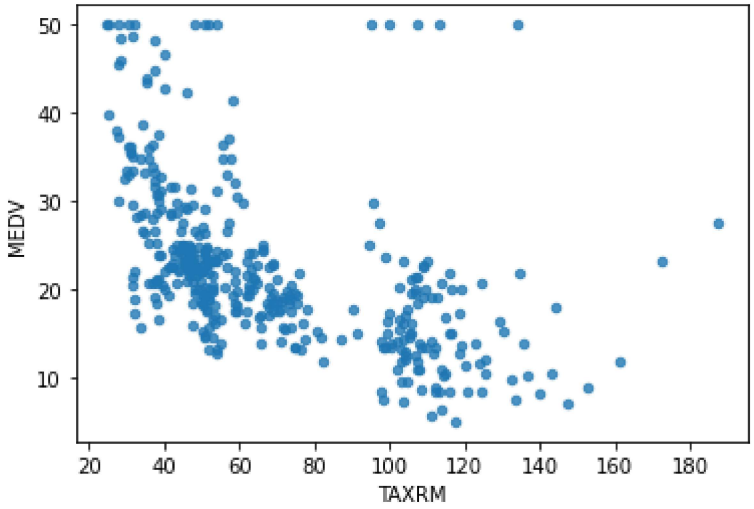
Out[91]:
```
MEDV       1.000000
RM         0.680857
B          0.361761
ZN         0.339741
DIS        0.240451
CHAS       0.205066
AGE       -0.364596
RAD       -0.374693
CRIM      -0.393715
NOX       -0.422873
TAX       -0.456657
INDUS     -0.473516
PTRATIO   -0.493534
TAXRM     -0.528626
LSTAT     -0.740494
Name: MEDV, dtype: float64
```

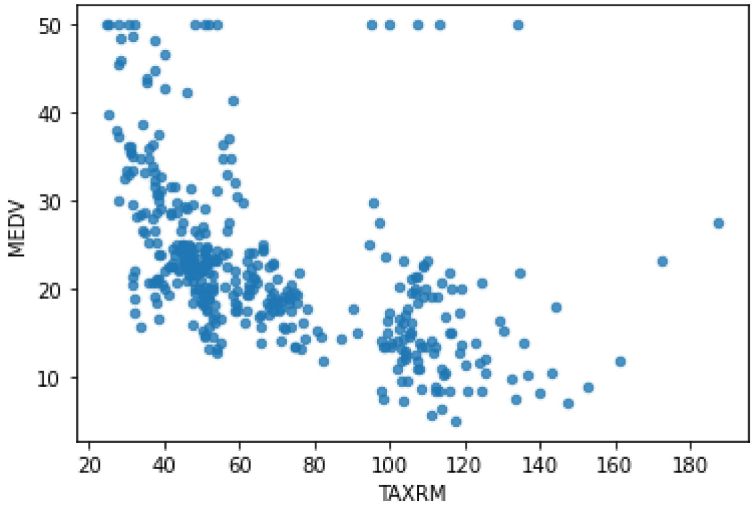
Out[91]:
```
MEDV       1.000000
RM         0.680857
B          0.361761
ZN         0.339741
DIS        0.240451
CHAS       0.205066
AGE       -0.364596
RAD       -0.374693
CRIM      -0.393715
NOX       -0.422873
TAX       -0.456657
INDUS     -0.473516
PTRATIO   -0.493534
TAXRM     -0.528626
LSTAT     -0.740494
Name: MEDV, dtype: float64
```

In [92]:
```
housing.plot(kind="scatter",x="TAXRM",y="MEDV",alpha=0.8)
```

Out[92]: <AxesSubplot:xlabel='TAXRM', ylabel='MEDV'>

Out[92]: <AxesSubplot:xlabel='TAXRM', ylabel='MEDV'>





In [93]:
```
housing=strat_train_set.drop("MEDV",axis=1)
housing_labels=strat_train_set["MEDV"].copy()
```

In [94]:
```
a=housing.dropna(subset=["RM"])
a.shape
```

Out[94]: (399, 13)

Out[94]: (399, 13)

In [95]:
```
housing.drop("RM",axis=1).shape
```

Out[95]: (404, 12)

Out[95]: (404, 12)

In [96]:
```
med=housing["RM"].median()
med
```

Out[96]: 6.209

Out[96]: 6.209

In [97]:
```
housing["RM"].fillna(med)
```

Out[97]:
```
254    6.108
348    6.635
476    6.484
321    6.376
326    6.312
       ...
155    6.152
423    6.103
98     7.820
455    6.525
216    5.888
Name: RM, Length: 404, dtype: float64
```

Out[97]:
```
254    6.108
348    6.635
476    6.484
321    6.376
326    6.312
       ...
155    6.152
423    6.103
98     7.820
455    6.525
216    5.888
Name: RM, Length: 404, dtype: float64
```

In [98]:
```
housing.shape
```

Out[98]: (404, 13)

Out[98]: (404, 13)

APPLYING IMPUTER

In [99]:
```
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(strategy="median")
imputer.fit(housing)
```

Out[99]: SimpleImputer(strategy='median')

Out[99]: SimpleImputer(strategy='median')

In [100]:
```
imputer.statistics_
```

Out[100]:
```
array([2.86735e-01, 0.00000e+00, 9.90000e+00, 0.00000e+00, 5.38000e-01,
       6.20900e+00, 7.82000e+01, 3.12220e+00, 5.00000e+00, 3.37000e+02,
       1.90000e+01, 3.90955e+02, 1.15700e+01])
```

Out[100]:
```
array([2.86735e-01, 0.00000e+00, 9.90000e+00, 0.00000e+00, 5.38000e-01,
       6.20900e+00, 7.82000e+01, 3.12220e+00, 5.00000e+00, 3.37000e+02,
       1.90000e+01, 3.90955e+02, 1.15700e+01])
```

In [101]:
```
x=imputer.transform(housing)
```

In [102]:
```
housing_tr=pd.DataFrame(x,columns=housing.columns)
```

In [103]:
```
housing_tr.describe()
```

Out[103]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.00
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.278609	69.039851	3.74
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.712366	28.258248	2.09
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.12
25%	0.086962	0.000000	5.190000	0.000000	0.453000	5.878750	44.850000	2.03
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.209000	78.200000	3.12
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.630000	94.100000	5.10
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

Out[103]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.00
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.278609	69.039851	3.74
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.712366	28.258248	2.09
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.12
25%	0.086962	0.000000	5.190000	0.000000	0.453000	5.878750	44.850000	2.03
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.209000	78.200000	3.12
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.630000	94.100000	5.10
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

In [104]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline=Pipeline([
    ('imputer',SimpleImputer(strategy="median")),
    ('std_scaler',StandardScaler()),
])
```

In [105]:
```
 housing_num_tr=my_pipeline.fit_transform(housing)
```

In [106]:
```
housing_num_tr.shape
```

Out[106]:
```
(404, 13)
```

Out[106]:
```
(404, 13)
```

In [107]:
```
housing_num_tr
```

Out[107]:
```
array([[-0.43942006,  3.12628155, -1.12165014, ..., -0.97491834,
         0.41164221, -0.86091034],
       [-0.44352175,  3.12628155, -1.35893781, ..., -0.69277865,
         0.39131918, -0.94116739],
       [ 0.15682292, -0.4898311 ,  0.98336806, ...,  0.81196637,
         0.44624347,  0.81480158],
       ...,
       [-0.43525657, -0.4898311 , -1.23083158, ..., -0.22254583,
         0.41831233, -1.27603303],
       [ 0.14210728, -0.4898311 ,  0.98336806, ...,  0.81196637,
        -3.15239177,  0.73869575],
       [-0.43974024, -0.4898311 ,  0.37049623, ..., -0.97491834,
         0.41070422,  0.09940681]])
```

Out[107]:
```
array([[-0.43942006,  3.12628155, -1.12165014, ..., -0.97491834,
         0.41164221, -0.86091034],
       [-0.44352175,  3.12628155, -1.35893781, ..., -0.69277865,
         0.39131918, -0.94116739],
       [ 0.15682292, -0.4898311 ,  0.98336806, ...,  0.81196637,
         0.44624347,  0.81480158],
       ...,
       [-0.43525657, -0.4898311 , -1.23083158, ..., -0.22254583,
         0.41831233, -1.27603303],
       [ 0.14210728, -0.4898311 ,  0.98336806, ...,  0.81196637,
        -3.15239177,  0.73869575],
       [-0.43974024, -0.4898311 ,  0.37049623, ..., -0.97491834,
         0.41070422,  0.09940681]])
```

SELECTING A DESIRED MODEL

In [108]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
model=LinearRegression()
model1=DecisionTreeRegressor()
model2=RandomForestRegressor()
model.fit(housing_num_tr,housing_labels)
```

Out[108]:
```
LinearRegression()
```

Out[108]:
```
LinearRegression()
```

In [109]:
```
model1.fit(housing_num_tr,housing_labels)
```

Out[109]:
```
DecisionTreeRegressor()
```

Out[109]:
```
DecisionTreeRegressor()
```

In [110]:
```
model2.fit(housing_num_tr,housing_labels)
```

Out[110]:
```
RandomForestRegressor()
```

Out[110]:
```
RandomForestRegressor()
```

In [111]:
```
some_data=housing.iloc[:5]
```

In [112]:
```
some_labels=housing_labels.iloc[:5]
```

In [113]:
```
prepared_data=my_pipeline.transform(some_data)
```

In [114]:
```
model2.predict(prepared_data)
```

Out[114]:
```
array([22.429, 25.767, 16.264, 23.24 , 23.696])
```

Out[114]:
```
array([22.429, 25.767, 16.264, 23.24 , 23.696])
```

In [115]:
```
list(some_labels)
```

Out[115]:
```
[21.9, 24.5, 16.7, 23.1, 23.0]
```

Out[115]:
```
[21.9, 24.5, 16.7, 23.1, 23.0]
```

In [116]:
```
from sklearn.metrics import mean_squared_error
housing_predictions=model2.predict(housing_num_tr)
mse=mean_squared_error(housing_labels,housing_predictions)
rmse=np.sqrt(mse)
```

In [117]:
```
rmse
```

Out[117]:
```
1.2328010186097134
```

Out[117]:
```
1.2328010186097134
```

In [118]:
```
from sklearn.model_selection import cross_val_score
scores=cross_val_score(model2,housing_num_tr,housing_labels,scoring="neg_mean_squ
rmse_scores=np.sqrt(-scores)
```

In [119]:
```
rmse_scores
```

Out[119]:
```
array([2.7068059 , 2.73108771, 4.45091152, 2.62686753, 3.4741107 ,
       2.72572932, 4.32888362, 3.27130495, 3.63729586, 3.36412864])
```

Out[119]:
```
array([2.7068059 , 2.73108771, 4.45091152, 2.62686753, 3.4741107 ,
       2.72572932, 4.32888362, 3.27130495, 3.63729586, 3.36412864])
```

In [120]:
```
def print_scores(scores):
    print("SCORES :",scores)
    print("MEAN :",scores.mean())
    print("STD__DEVIATION :",scores.std())
```

In [121]:
```
print_scores(rmse_scores)
```

```
SCORES : [2.7068059  2.73108771 4.45091152 2.62686753 3.4741107  2.72572932
 4.32888362 3.27130495 3.63729586 3.36412864]
MEAN : 3.3317125739977236
STD__DEVIATION : 0.6309196027863562
SCORES : [2.7068059  2.73108771 4.45091152 2.62686753 3.4741107  2.72572932
 4.32888362 3.27130495 3.63729586 3.36412864]
MEAN : 3.3317125739977236
STD__DEVIATION : 0.6309196027863562
```

In [122]:
```
from joblib import dump,load
dump(model2,'Dragon.joblib')
```

Out[122]:
```
['Dragon.joblib']
```

Out[122]:
```
['Dragon.joblib']
```

TESTING

In [123]:
```
x_test=strat_test_set.drop("MEDV",axis=1)
y_test=strat_test_set["MEDV"].copy()
x_test_prepared=my_pipeline.transform(x_test)
final_predictions=model.predict(x_test_prepared)
final_mse=mean_squared_error(y_test,final_predictions)
final_rmse=np.sqrt(final_mse)
```

In [124]:
```
final_rmse
```

Out[124]:
```
4.143874870573364
```

Out[124]:
```
4.143874870573364
```

In [125]:
```
prepared_data[1]
```

Out[125]:
```
array([-0.44352175,  3.12628155, -1.35893781, -0.27288841, -1.0542567 ,
        0.5009123 , -1.3938808 ,  2.19312325, -0.65766683, -0.78557904,
       -0.69277865,  0.39131918, -0.94116739])
```

Out[125]:
```
array([-0.44352175,  3.12628155, -1.35893781, -0.27288841, -1.0542567 ,
        0.5009123 , -1.3938808 ,  2.19312325, -0.65766683, -0.78557904,
       -0.69277865,  0.39131918, -0.94116739])
```

USING THE MODEL

In [126]:
```
from joblib import dump,load
import numpy as np
model=load('Dragon.joblib')
features=np.array([[ -1.44352175,  18.12628155, -10.35893781, -0.27288841, -
        0.5009123 , -10.3938808 ,  12.19312325, -0.65766683, -0.78557904,
       -0.69277865,  33.39131918, -0.94116739  ]])
model.predict(features)
```

Out[126]:
```
array([25.992])
```

Out[126]:
```
array([25.992])
```

In [ ]:

In [ ]: