House-Price-Predictor

In [1]:
```python
import pandas as pd
```

In [2]:
```python
housing=pd.read_csv("data.csv")
```

In [3]:
```python
housing.head()
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	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	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	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	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	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	18.7	396.90	5.33	36.2

In [4]:
```python
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       506 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 [5]:
```python
housing["CHAS"].value_counts()
```

Out[5]:
```
0    471
1     35
Name: CHAS, dtype: int64
```

In [6]:
```python
housing.describe()
```

Out[6]:

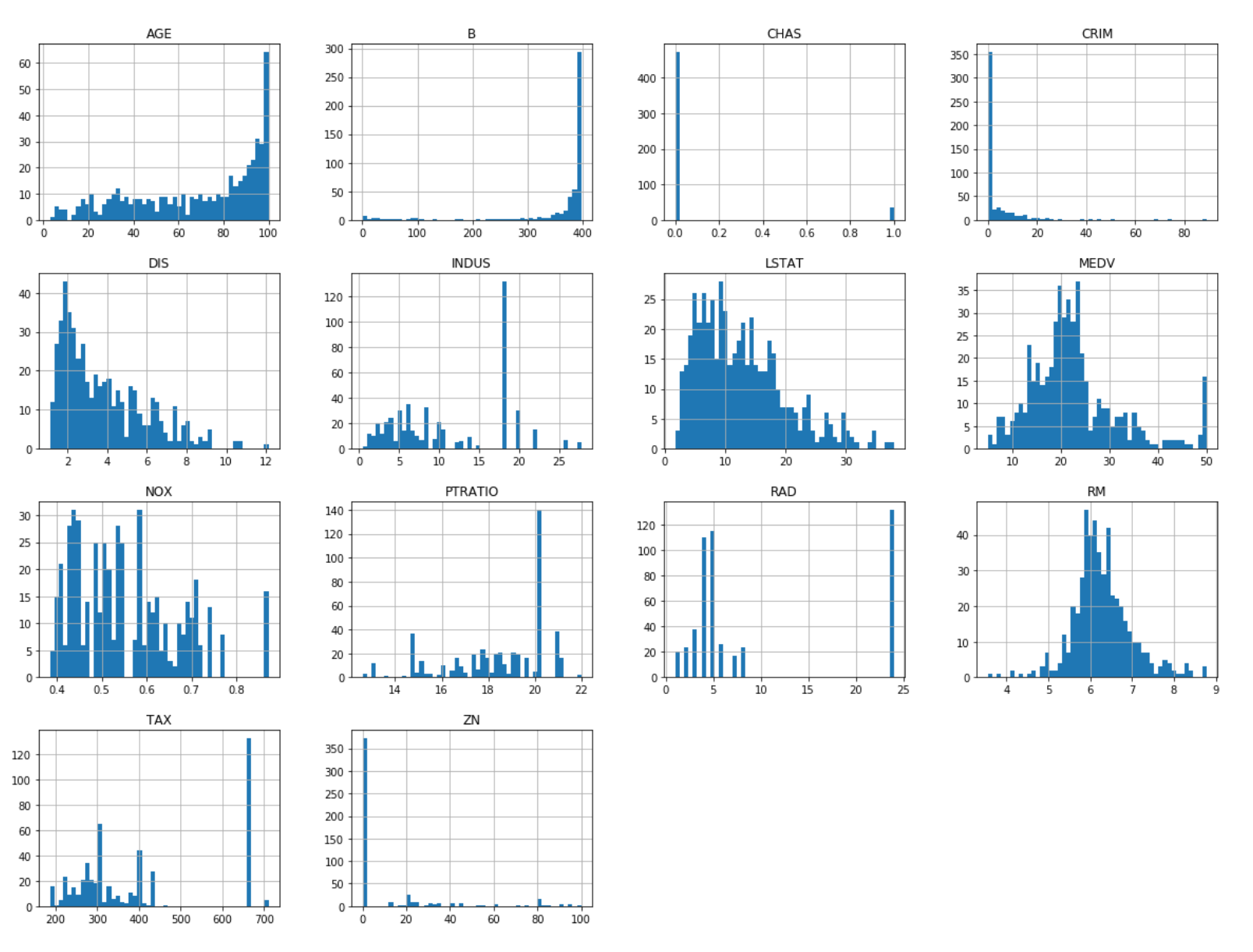
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TA
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.23715
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.53711
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.00000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.00000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.00000
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.00000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.00000

In [7]:
```python
%matplotlib inline
```

In [8]:
```python
import matplotlib.pyplot as plt
```

In [9]:
```python
housing.hist(bins=50,figsize=(20,15))
```

Out[9]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B646ACDC0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B6493E850>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64978E80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B649A1940>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B649D8F70>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64A00A00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64A0C910>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64A421F0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64A95340>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64ABCB80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64AF0430>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64B19C70>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64B4E4F0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64B77D30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64BBA3A0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B64BBA640>]],
      dtype=object)
```



Train Test Splitting

In [10]:
```python
import numpy as np
def split_train_test(data,test_ratio):
    np.random.seed(42)
    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 [11]:
```python
train_set,test_set=split_train_test(housing,0.2)
```

In [12]:
```python
print(len(test_set))
```

```
101
```

In [13]:
```python
from sklearn.model_selection import train_test_split
train_set,test_set=train_test_split(housing,test_size=0.2,random_state=42)
```

In [14]:
```python
len(train_set)
```

Out[14]:
```
404
```

In [15]:
```python
len(test_set)
```

Out[15]:
```
102
```

In [16]:
```python
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 [17]:
```python
strat_test_set['CHAS'].value_counts()
```

Out[17]:
```
0    95
1     7
Name: CHAS, dtype: int64
```

In [18]:
```python
housing=strat_train_set.copy()
```

Looking for correlations`

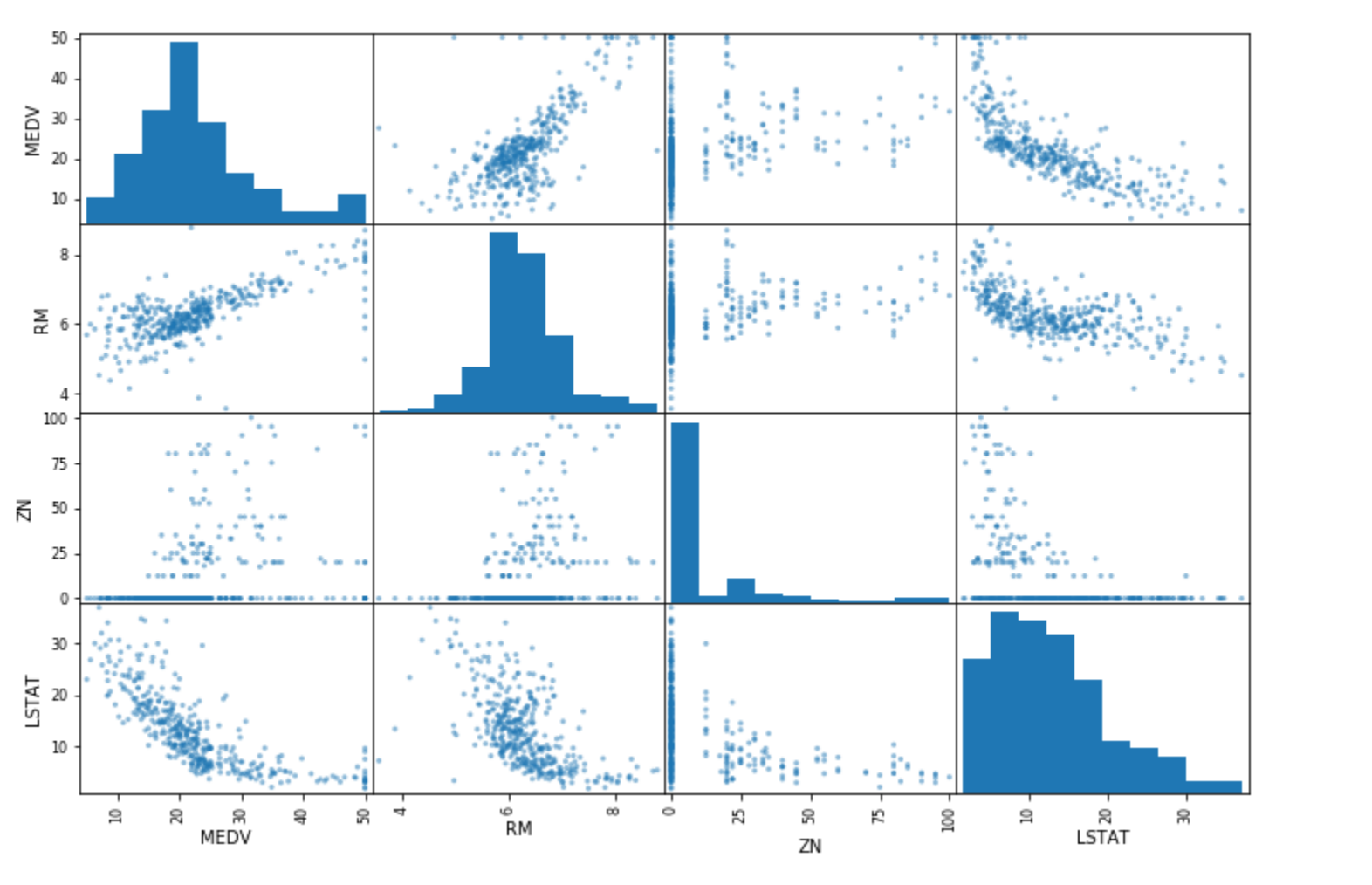
In [19]:
```python
corr_matrix=housing.corr()
```

In [20]:
```python
corr_matrix['MEDV'].sort_values(ascending=False)
```

Out[20]:
```
MEDV       1.000000
RM         0.679894
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
```

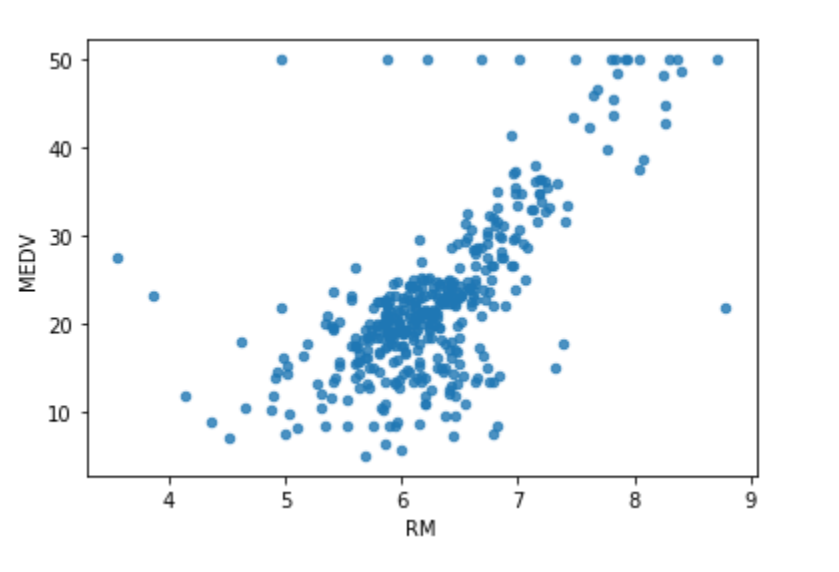
In [21]:
```python
from pandas.plotting import scatter_matrix
attributes=["MEDV","RM","ZN","LSTAT"]
scatter_matrix(housing[attributes],figsize=(12,8))
```

Out[21]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67888310>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B6789D820>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B678D8F40>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B6790B7C0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67943DF0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B6796B880>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67977790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B679B0070>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67A01F70>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67A2AA00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67A5F280>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67A88AC0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67ABE340>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67AE8B80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67B1D400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029B67B46C40>]],
      dtype=object)
```



In [22]:
```python
housing.plot(kind="scatter",x="RM",y="MEDV",alpha=0.8)
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29b67f01a30>
```



Trying out Atribute Combinations

In [23]:
```python
housing['TAXRM']=housing['TAX']/housing['RM']
housing.head()
```

Out[23]:

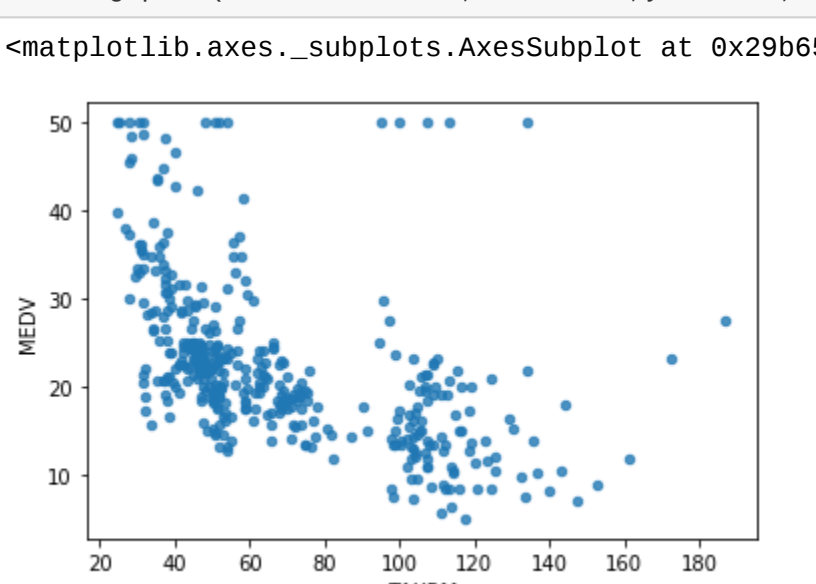
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	TAXRM
254	0.04819	80.0	3.64	0	0.392	6.108	32.0	9.2203	1	315	16.4	392.89	6.57	21.9	51.571709
348	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	390.94	5.99	24.5	42.200452
476	4.87141	0.0	18.10	0	0.614	6.484	93.6	2.3053	24	666	20.2	396.21	18.68	16.7	102.714374
321	0.18159	0.0	7.38	0	0.493	6.376	54.3	4.5404	5	287	19.6	396.90	6.87	23.1	45.012547
326	0.30347	0.0	7.38	0	0.493	6.312	28.9	5.4159	5	287	19.6	396.90	6.15	23.0	45.468948

In [24]:
```python
corr_matrix=housing.corr()
corr_matrix['MEDV'].sort_values(ascending=False)
```

Out[24]:
```
MEDV       1.000000
RM         0.679894
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     -0.525160
LSTAT     -0.740494
Name: MEDV, dtype: float64
```

In [25]:
```python
housing.plot(kind="scatter",x="TAXRM",y="MEDV",alpha=0.8)
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29b65b4d4f0>
```



In [26]:
```python
housing=strat_train_set.drop("MEDV",axis=1)
housing_labels=strat_train_set["MEDV"].copy()
```

In [27]:
```python
housing.describe()
```

Out[27]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TA
count	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.00000
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.279908	69.039851	3.746210	9.735149	412.34158
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.712983	28.258248	2.099057	8.731259	168.67262
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.129600	1.000000	187.00000
25%	0.086963	0.000000	5.190000	0.000000	0.453000	5.878750	44.850000	2.035975	4.000000	284.00000
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.210000	78.200000	3.122200	5.000000	337.00000
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.630250	94.100000	5.100400	24.000000	666.00000
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.00000

Creating a Pipeline

In [28]:
```python
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(strategy="median")
imputer.fit(housing)
```

Out[28]:
```
SimpleImputer(add_indicator=False, copy=True, fill_value=None,
              missing_values=nan, strategy='median', verbose=0)
```

In [29]:
```python
imputer.statistics_
```

Out[29]:
```
array([2.86735e-01, 0.00000e+00, 9.90000e+00, 0.00000e+00, 5.38000e-01,
       6.21000e+00, 7.82000e+01, 3.12220e+00, 5.00000e+00, 3.37000e+02,
       1.90000e+01, 3.90955e+02, 1.15700e+01])
```

In [30]:
```python
x=imputer.transform(housing)
```

In [31]:
```python
housing_tr=pd.DataFrame(x,columns=housing.columns)
```

In [32]:
```python
housing_tr.describe()
```

Out[32]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TA
count	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.00000
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.279908	69.039851	3.746210	9.735149	412.34158
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.712983	28.258248	2.099057	8.731259	168.67262
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.129600	1.000000	187.00000
25%	0.086963	0.000000	5.190000	0.000000	0.453000	5.878750	44.850000	2.035975	4.000000	284.00000
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.210000	78.200000	3.122200	5.000000	337.00000
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.630250	94.100000	5.100400	24.000000	666.00000
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.00000

In [33]:
```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline=Pipeline([('imputer',SimpleImputer(strategy='median')),
                      ('std_scaler',StandardScaler())
                      ])
```

In [34]:
```python
housing_num_tr=my_pipeline.fit_transform(housing)
```

In [35]:
```python
housing_num_tr.shape
```

Out[35]:
```
(404, 13)
```

Selecting a Desired Model

In [36]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
#model=DecisionTreeRegressor()
#model=LinearRegression()
model=RandomForestRegressor()
model.fit(housing_num_tr,housing_labels)
```

Out[36]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, 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=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [37]:
```python
some_data=housing.iloc[:5]
some_labels=housing_labels.iloc[:5]
```

In [38]:
```python
prepared_data=my_pipeline.transform(some_data)
model.predict(prepared_data)
```

Out[38]:
```
array([22.527, 25.535, 16.229, 23.363, 23.387])
```

In [39]:
```python
list(some_labels)
```

Out[39]:
```
[21.9, 24.5, 16.7, 23.1, 23.0]
```

Evaluating The Model

In [40]:
```python
from sklearn.metrics import mean_squared_error
housing_predictions=model.predict(housing_num_tr)
mse=mean_squared_error(housing_labels,housing_predictions)
rmse=np.sqrt(mse)
```

In [41]:
```python
rmse
```

Out[41]:
```
1.1630122644356427
```

using better evaluation technique-cross validation

In [42]:
```python
from sklearn.model_selection import cross_val_score
scores=cross_val_score(model,housing_num_tr,housing_labels,scoring="neg_mean_squared_error",
cv=10)
rmse_scores=np.sqrt(-scores)
```

In [43]:
```python
rmse_scores
```

Out[43]:
```
array([2.80771915, 2.70494939, 4.39952315, 2.55603067, 3.32453776,
       2.53247315, 4.79801114, 3.27205951, 3.28241181, 3.19348244])
```

In [44]:
```python
def print_scores(scores):
    print("Scores:",scores)
    print("Mean:",scores.mean())
    print("Standard deviation",scores.std())
```

In [45]:
```python
print_scores(rmse_scores)
```

```
Scores: [2.80771915 2.70494939 4.39952315 2.55603067 3.32453776 2.53247315
 4.79801114 3.27205951 3.28241181 3.19348244]
Mean: 3.287119816217655
Standard deviation 0.7213446572200697
```

In [46]:
```python
from joblib import dump,load
dump(model,'house_price.joblib')
```

Out[46]:
```
['house_price.joblib']
```

In [49]:
```python
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)
print(final_predictions,list(y_test))
```

```
[25.083 11.592 25.545 22.016 18.384 15.108 19.992 14.368 31.892 40.393
 19.748 11.769 24.037 28.724 19.433 10.517 31.876 14.482 23.579 18.761
 19.641 18.082 17.54  22.078 18.32  30.47  16.266 32.763  8.897 33.838
 24.008 21.246 23.088 10.994 20.876 11.264 42.506 24.36  23.293 41.51
 23.83  29.639 20.457 20.906 19.263 33.644 44.494 19.88  20.165 21.769
 21.512 14.708 21.32  15.086 24.762 32.839 42.412 28.135 19.397 20.85
 47.428 10.17  18.535 24.606 15.002 33.016 19.463 17.786 19.002 33.884
 27.242 22.853 21.469 22.538 35.259 12.64  15.839 20.006 20.689 21.375
 22.331 21.671 14.175 22.861 20.876 21.135 13.944 21.295 22.067 23.166
 18.956 27.224  7.258 25.967 18.556 29.995 19.64  31.196 14.616 26.521
 20.82  20.092] [16.5, 10.2, 30.1, 23.0, 14.4, 15.6, 19.4, 14.1, 30.3, 35.2, 23.1, 13.8, 25.
0, 27.9, 19.5, 12.3, 32.2, 13.5, 23.8, 21.7, 19.2, 19.5, 10.4, 23.2, 18.6, 28.5, 15.2, 32.0,
7.2, 34.6, 20.1, 20.6, 23.6, 13.1, 23.8, 12.7, 43.1, 24.7, 22.2, 44.0, 28.1, 31.0, 21.7, 23.
4, 19.5, 33.1, 41.7, 18.7, 19.9, 20.6, 21.2, 13.6, 20.3, 17.8, 27.1, 31.5, 50.0, 29.1, 18.9,
20.4, 50.0, 7.2, 17.2, 36.2, 14.6, 33.2, 23.8, 19.9, 21.5, 37.3, 27.0, 22.0, 24.3, 19.8, 33.
3, 7.0, 19.4, 20.9, 21.1, 20.4, 22.2, 11.9, 11.7, 21.6, 19.7, 23.0, 16.7, 21.7, 20.6, 23.3, 1
9.6, 28.0, 5.0, 24.4, 20.8, 24.8, 21.8, 23.6, 19.0, 25.0, 20.3, 21.5]
```

In [48]:
```python
final_rmse
```

Out[48]:
```
2.961858053224525
```

In [ ]: