REAL STATE PRICE PRIDICTER

In [65]:
```
import pandas as pd
```

In [66]:
```
housing = pd.read_csv("data.csv")
```

In [67]:
```
housing.head()
```

Out[67]:

	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 [68]:
```
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
```

In [69]:
```
housing.describe()
```

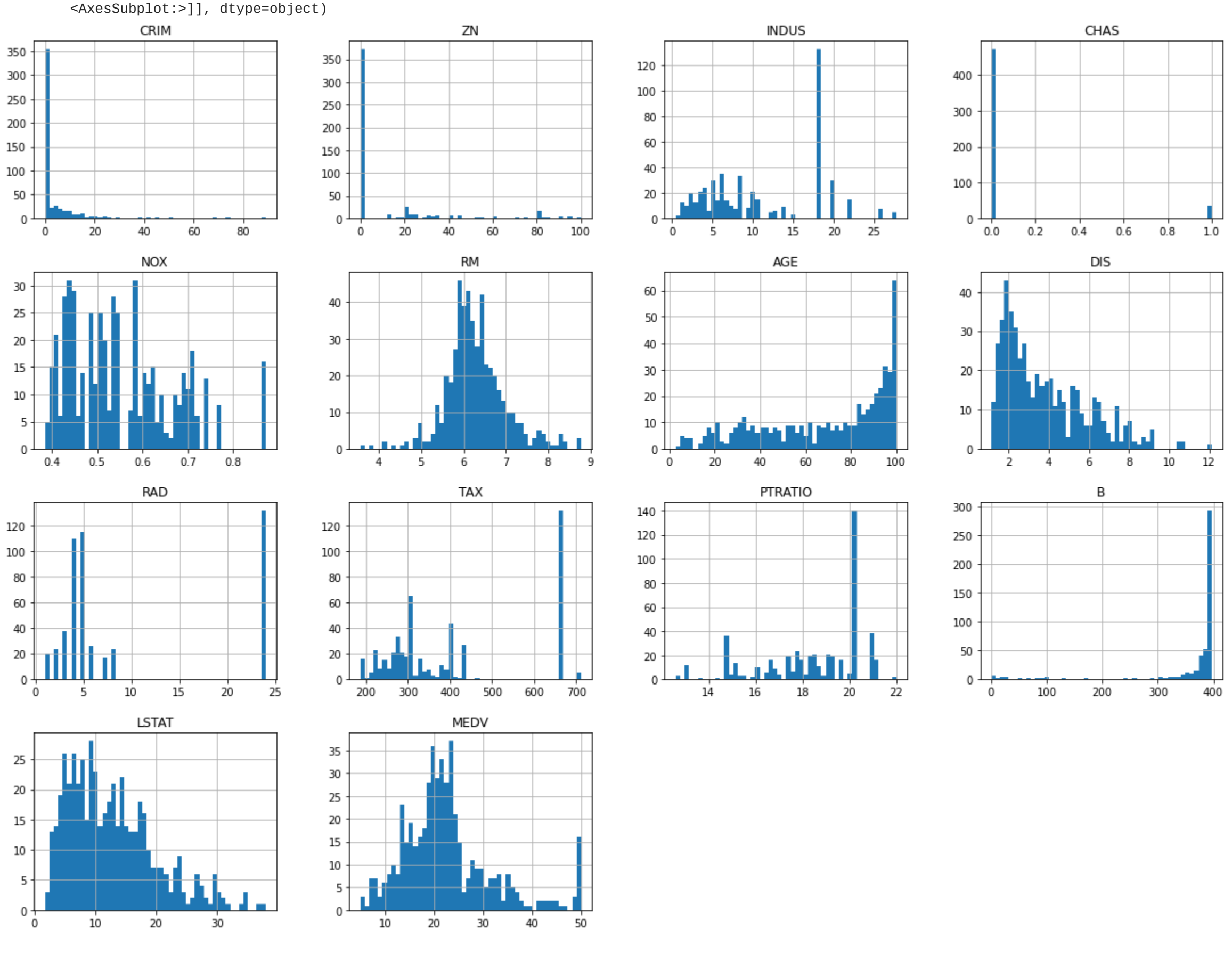
Out[69]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
count	506.000000	506.000000	506.000000	506.000000	506.000000	501.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.287096	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.705510	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885000	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.209000	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.629000	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

In [70]:
```
%matplotlib inline
```

In [71]:
```
#for ploting histogram
import matplotlib.pyplot as plt
housing.hist(bins=50 , figsize=(20,15)) #we have to write "plt.show" in the end to see the graphs if we are coading any ID (PYCHARM ETC.)
```

Out[71]:
```
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)
```



TRAIN-TEST SPLITING

In [72]:
```
#for learning purpose


# import numpy as np

# def split_train_test(data, test_ratio):
#     np.random.seed(42)#if dont use the "seed" then random fun. will give diff. test,train data everytime we run it  due to which some test data will be seen by algo before testin
#     shuffled = np.random.permutation(len(data))
#     print(shuffled)
#     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 [73]:
```
#  train_set, test_set = split_train_test(housing, 0.2)
```

In [74]:
```
# print(f"rows in train set: {len(train_set)} \nrows in test set: {len(test_set)}\n")
```

In [75]:
```
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
```

In [76]:
```
#Stratified Shuffing Split = it insure that model should get every type of data so that it can train accordingly(exam. =  CHAS ki value 1 bhi hosakti he ye to tumne batya hi nahi t
```

In [77]:
```
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 [78]:
```
 strat_train_set["CHAS"].value_counts()
```

Out[78]:
```
0    376
1     28
Name: CHAS, dtype: int64
```

In [79]:
```
 strat_test_set["CHAS"].value_counts()
```

Out[79]:
```
0    95
1     7
Name: CHAS, dtype: int64
```

In [80]:
```
#95/7 = 376/28
```

In [81]:
```
housing = strat_train_set.copy()#ye mene imupter k baad kiya
#while working with big data in ML iss step k baad ik copy lele
```

looking for correlation

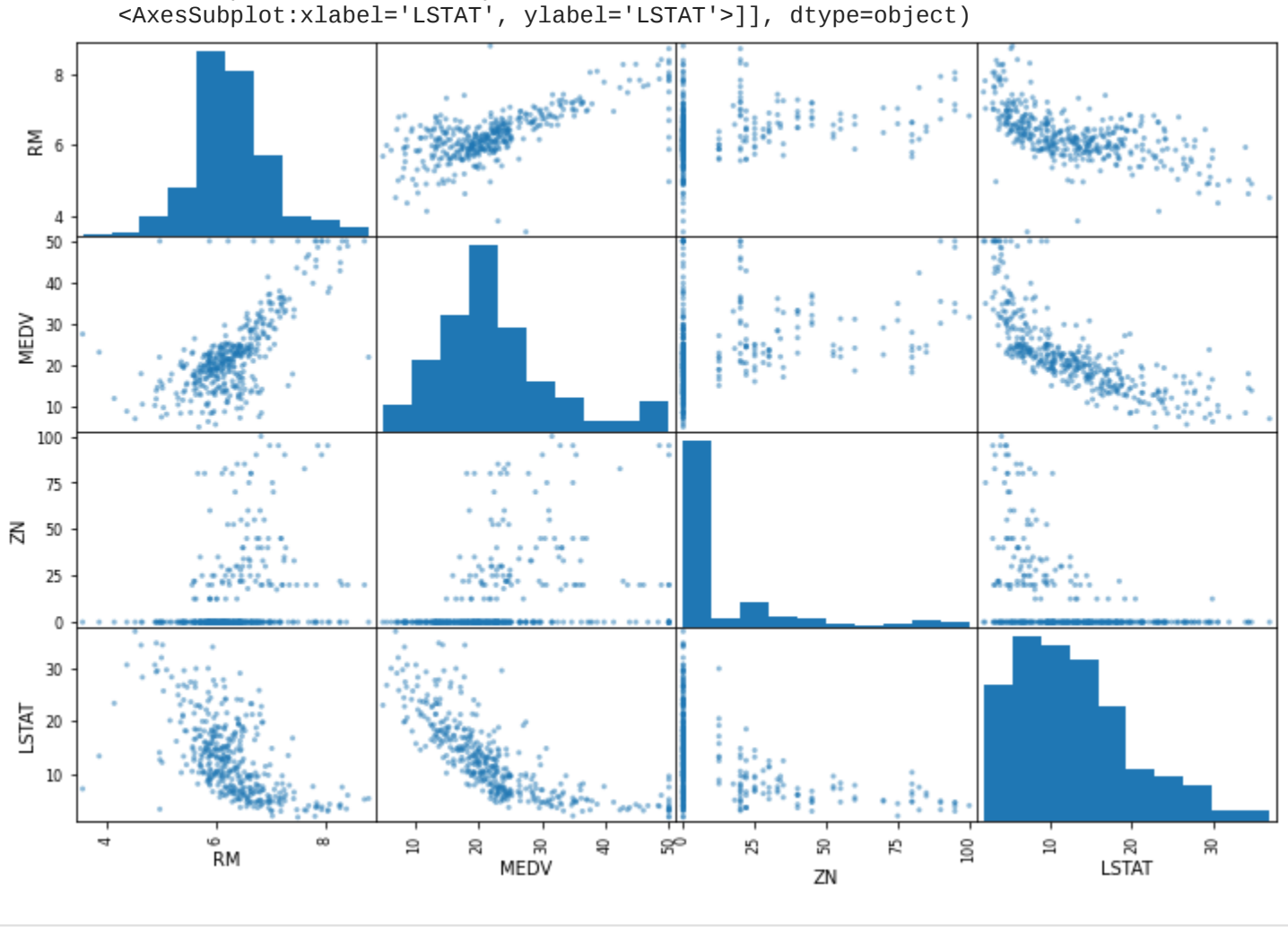
In [82]:
```
corr_matrix = housing.corr()
corr_matrix["MEDV"].sort_values(ascending=False)
```

Out[82]:
```
MEDV       1.000000
RM         0.679933
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 [83]:
```
#pearson correlation coff = dependence of one thing on dec. , inc. of other thing
# (kisi chij ko  agar inc. ya dec. karoge to dosri chij inc hogi ya dec.)
# strong + correlation =1 ( ye inc. to wo bhi inc. same rate se)
# strong - correlation =-1 ( ye inc. to wo dec. same rate se)
```

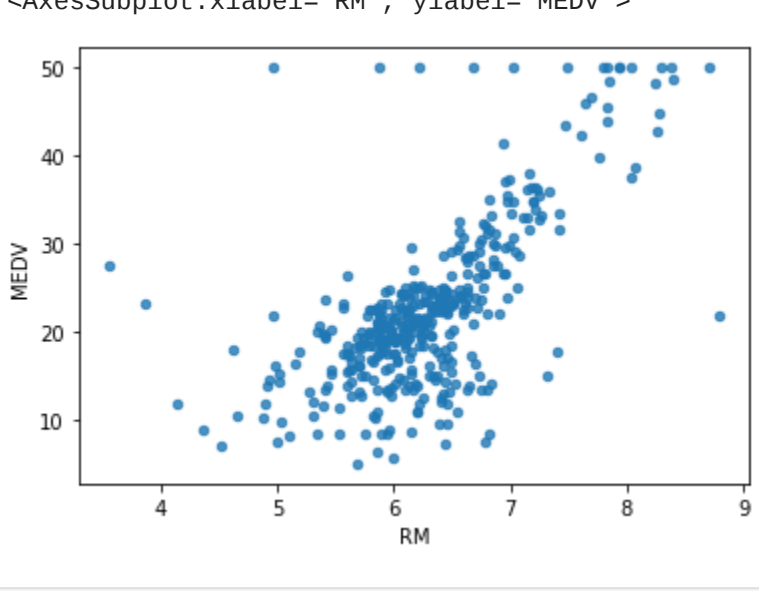
In [84]:
```
from pandas.plotting import scatter_matrix
attributes = ["RM","MEDV","ZN","LSTAT"]
scatter_matrix(housing[attributes], figsize = (12,8))
```

Out[84]:
```
array([[<AxesSubplot:xlabel='RM', ylabel='RM'>,
        <AxesSubplot:xlabel='MEDV', ylabel='RM'>,
        <AxesSubplot:xlabel='ZN', ylabel='RM'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='RM'>],
       [<AxesSubplot:xlabel='RM', ylabel='MEDV'>,
        <AxesSubplot:xlabel='MEDV', ylabel='MEDV'>,
        <AxesSubplot:xlabel='ZN', ylabel='MEDV'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='MEDV'>],
       [<AxesSubplot:xlabel='RM', ylabel='ZN'>,
        <AxesSubplot:xlabel='MEDV', ylabel='ZN'>,
        <AxesSubplot:xlabel='ZN', ylabel='ZN'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='ZN'>],
       [<AxesSubplot:xlabel='RM', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='MEDV', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='ZN', ylabel='LSTAT'>,
        <AxesSubplot:xlabel='LSTAT', ylabel='LSTAT'>]], dtype=object)
```



In [85]:
```
housing.plot(kind = "scatter", x="RM", y="MEDV" , alpha=0.8)
```

Out[85]:
```
<AxesSubplot:xlabel='RM', ylabel='MEDV'>
```



In [86]:
```
#hum outlaiers ko hata sakte he (outlers matlab ju point bahar wale he)
```

trying out attributes combination

In [87]:
```
#we can use two attributes and create new one , this can help in improving training module
```

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
```

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

Out[90]:

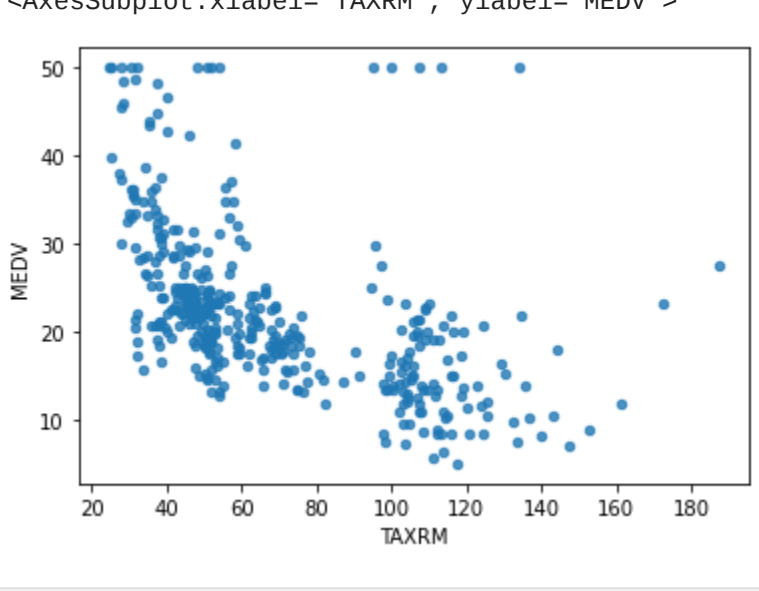
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	TAXRM
254	0.04819	80.0	3.64	0	0.392	6.108	32.0	9.2203	1	315	16.4	392.89	6.57	21.9	51.571709
348	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	390.94	5.99	24.5	42.200452
476	4.87141	0.0	18.10	0	0.614	6.484	93.6	2.3053	24	666	20.2	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 [91]:
```
corr_matrix = housing.corr()
corr_matrix["MEDV"].sort_values(ascending=False)
```

Out[91]:
```
MEDV       1.000000
RM         0.679933
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     -0.526094
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'>
```



In [93]:
```
#ye mene selecting ml k time paar banya tha
housing = strat_train_set.drop('MEDV', axis =1)
housing_labels = strat_train_set["MEDV"].copy()
```

missing attributes

In [94]:
```
#to take care of missing attributes ,  you have three option:
#  1.get rid of the missing data point
#  2.get rid of the whole attributes
#  3.set the value to same value(0, mean or medium)
```

In [95]:
```
#1agar ik do missing data hute he to hata dete he agar jayda missing he to nahi
#2agar correltaion 0 k near huta to hata dete but "RM" nahi he 0 k near
#
```

In [96]:
```
#1 execution
a=housing.dropna(subset=["RM"]).shape
```

In [97]:
```
#2 execution
housing.drop("RM", axis=1).shape
#there is no RM coloumn and also note that original housing dataframe will remain unchanged
```

Out[97]:
```
(404, 12)
```

In [98]:
```
#3 execution  compute median
median = housing["RM"].median()
```

In [99]:
```
median
```

Out[99]:
```
6.2135
```

In [100…
```
#3
housing["RM"].fillna(median)
# note that original housing dataframe will remain unchanged
```

Out[100…
```
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
```

agar kahi bhi missing value hogi data set me ya fir input me waha ye median

bhar dena (niche)

In [101…
```
housing.describe()#before we started filling missing attributes
```

Out[101…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
count	404.000000	404.000000	404.000000	404.000000	404.000000	400.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.282595	69.039851	3.746210	9.735149	412.341584	18.473267	353.392822	12.791609
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.715845	28.258248	2.099057	8.731259	168.672623	2.129243	96.069235	7.235740
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.129600	1.000000	187.000000	13.000000	0.320000	1.730000
25%	0.086962	0.000000	5.190000	0.000000	0.453000	5.878750	44.850000	2.035975	4.000000	284.000000	17.400000	374.617500	6.847500
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.213500	78.200000	3.122200	5.000000	337.000000	19.000000	390.955000	11.570000
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.632000	94.100000	5.100400	24.000000	666.000000	20.200000	395.630000	17.102500
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	36.980000

In [102…
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
imputer.fit(housing)
```

Out[102…
```
SimpleImputer(strategy='median')
```

In [103…
```
imputer.statistics_
```

Out[103…
```
array([2.86735e-01, 0.00000e+00, 9.90000e+00, 0.00000e+00, 5.38000e-01,
       6.21350e+00, 7.82000e+01, 3.12220e+00, 5.00000e+00, 3.37000e+02,
       1.90000e+01, 3.90955e+02, 1.15700e+01])
```

In [104…
```
x=imputer.transform(housing)
```

In [105…
```
housing_tr = pd.DataFrame(x,columns=housing.columns)
```

In [106…
```
housing_tr.describe()
```

Out[106…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
count	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.281911	69.039851	3.746210	9.735149	412.341584	18.473267	353.392822	12.791609
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.712316	28.258248	2.099057	8.731259	168.672623	2.129243	96.069235	7.235740
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.129600	1.000000	187.000000	13.000000	0.320000	1.730000
25%	0.086962	0.000000	5.190000	0.000000	0.453000	5.879750	44.850000	2.035975	4.000000	284.000000	17.400000	374.617500	6.847500
50%	0.286735	0.000000	9.900000	0.000000	0.538000	6.213500	78.200000	3.122200	5.000000	337.000000	19.000000	390.955000	11.570000
75%	3.731923	12.500000	18.100000	0.000000	0.631000	6.630250	94.100000	5.100400	24.000000	666.000000	20.200000	395.630000	17.102500
max	73.534100	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	36.980000

scikit-learn design

primarly, three types of objects

1. estimators - It estimates some parameter based on a dataset. Eg. imputer . It has fit method and transform method. Fit method - fits the datasets and calculates internal parameters. it has some hyperparameters like strategy

2. transformers - transform method takes input an returns output based on the learning from fit(). it has a convenience function called fit_trandform() which fits and then transform

3. predictors - Linerregression model is an example of predictor. fit() and predict() are two common functions. it also gives score function which will evaluate the predictions

Feature Scaling

primarly , two types of feature scaling method:

1. Min-Max scaling (Normalisation) (value-min)/(max-min) sklearn provide a class called MinMaxScaler for this

2. Stadardization (Value - Mean)/std Sklearn provides a class called StandardScaler for this

creating pipeline

In [107…
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy='median')),
    #......add as many as you want in your pipeline
    ('std_scaler', StandardScaler())
])
```

In [108…
```
housing_num_tr = my_pipeline.fit_transform(housing)
```

In [109…
```
housing_num_tr
```

Out[109…
```
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]])
```

In [110…
```
housing_num_tr.shape
```

Out[110…
```
(404, 13)
```

yaha hum pipeline alag bana rahe he but real world problem me pipeline starting se hi banana huta he

selecting a desired model for Real Estates

In [111…
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
# model = LinearRegression() # error bhut jayda tha , cross validation k baad bhi
model = DecisionTreeRegressor() #overfitting karliya , isiliye cross validation kardiya
# model= RandomForestRegressor()
model.fit(housing_num_tr,housing_labels)
```

Out[111…
```
DecisionTreeRegressor()
```

In [112…
```
some_data = housing.iloc[:5]
```

In [113…
```
some_labels = housing_labels.iloc[:5]
```

In [114…
```
prepared_data = my_pipeline.transform(some_data)
```

In [115…
```
model.predict(prepared_data)
```

Out[115…
```
array([21.9, 24.5, 16.7, 23.1, 23. ])
```

In [116…
```
some_labels
```

Out[116…
```
254    21.9
348    24.5
476    16.7
321    23.1
326    23.0
Name: MEDV, dtype: float64
```

evaluating the model

In [117…
```
import numpy as np
from sklearn.metrics import mean_squared_error
housing_prediction = model.predict(housing_num_tr)
mse = mean_squared_error (housing_labels,housing_prediction )
rmse= np.sqrt(mse)
```

In [118…
```
mse
```

Out[118…
```
0.0
```

using better evaluation technique - cross validation

k= 10 (1 2 3 4 5 6 7 8 9 10) (ik ko chur k sabb pe train karo fir jisse chura usse test , aesae sabke sath karna he fir ju sabka error ayega usse use karna he)

In [119…
```
from sklearn.model_selection import cross_val_score
scores =cross_val_score(model ,housing_num_tr,housing_labels , scoring= "neg_mean_squared_error" )
rmse_score =np.sqrt(-scores)
```

In [120…
```
rmse_score
```

Out[120…
```
array([5.01012555, 4.22365473, 5.19992877, 5.27708767, 3.54282726])
```

In [121…
```
def print_scores(scores):
    print("scores", scores)
    print("mean",scores.mean())
    print("standard devation", scores.std())
```

In [122…
```
print_scores(rmse_score)
```

```
scores [5.01012555 4.22365473 5.19992877 5.27708767 3.54282726]
mean 4.650724797397023
standard devation 0.668273919673598
```

saving the model

In [123…
```
from joblib import dump,load
dump(model, 'Dragon.joblib')
```

Out[123…
```
['Dragon.joblib']
```

testing the model on test data

In [124…
```
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_prediction = model.predict(x_test_prepared)
final_mse = mean_squared_error(y_test ,final_prediction)
final_rmse = np.sqrt(final_mse)
# print(final_prediction ,list(y_test) )
```

In [125…
```
final_rmse
```

Out[125…
```
4.1374189169236475
```

In [126…
```
prepared_data[0]
```

Out[126…
```
array([-0.43942006,  3.12628155, -1.12165014, -0.27288841, -1.42262747,
       -0.2444511 , -1.31238772,  2.61111401, -1.0016859 , -0.5778192 ,
       -0.97491834,  0.41164221, -0.86091034])
```

using the module

In [127…
```
from joblib import dump,load
import numpy as np
model= load( 'Dragon.joblib')

features = np.array([[-0.43942006,  3.12628155, -1.12165014, -0.27288841, -1.42262747,
        -0.2444511 , -1.31238772,  2.61111401, -1.0016859 , -0.5778192 ,
        -0.97491834,  0.41164221, -0.86091034]])
model.predict(features)
```

Out[127…
```
array([21.9])
```

kai baar package update hojate he isiliye warning aate ge , same with sklearn to error ko read karo , google search karo