In [1]:
```
import pandas as pd
import numpy as np
```

Reading Data

In [2]:
```
df=pd.read_csv('E:\Project\Prediction\data.csv')
df.head(5)
```

Out[2]:

	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 [3]:
```
df.isnull().sum()
```

Out[3]:
```
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 [4]:
```
df['CHAS'].value_counts()
```

Out[4]:
```
0    471
1     35
Name: CHAS, dtype: int64
```

In [5]:
```
df.fillna(df.mean(),inplace=True)
df.isnull().sum()
```

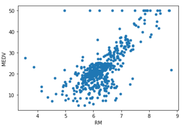
Out[5]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [6]:
```
df.shape
```

Out[6]:
```
(506, 14)
```

In [7]:
```
df.plot(kind='scatter',x='RM',y='MEDV',alpha=1)
```

Out[7]:
```
<AxesSubplot:xlabel='RM', ylabel='MEDV'>
```



Train-Test Splitting

In [8]:
```
from sklearn.model_selection import train_test_split
train_set,test_set= train_test_split(df,random_state=42,test_size=0.2)
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
```

Shuffle-Split

In [9]:
```
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(df,df['CHAS']):
    strat_test_set=df.iloc[test_index]
    strat_train_set=df.iloc[train_index]
```

In [10]:
```
strat_test_set['CHAS'].value_counts()
```

Out[10]:
```
0    95
1     7
Name: CHAS, dtype: int64
```

In [11]:
```
strat_train_set['CHAS'].value_counts()
```

Out[11]:
```
0    376
1     28
Name: CHAS, dtype: int64
```

In [12]:
```
dftt=strat_train_set.drop(['MEDV'],axis=1)
dftt_label=strat_train_set['MEDV']
```

In [13]:
```
dftt_label.head(5)
```

Out[13]:
```
254    21.9
348    24.5
476    16.7
321    23.1
326    23.0
Name: MEDV, dtype: float64
```

In [14]:
```
dftt.shape
```

Out[14]:
```
(404, 13)
```

Creating Pipeline

In [15]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipe=Pipeline([
    ('std_scaler',StandardScaler())
])
```

In [16]:
```
dftt_pp=my_pipe.fit_transform(dftt)
dftt_pp.shape
```

Out[16]:
```
(404, 13)
```

Using Algorithm

In [17]:
```
#from sklearn.linear_model import LinearRegression
#from sklearn.tree import DecisionTreeRegressor
#from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
#model=LinearRegression()
#model=DecisionTreeRegressor()
#model=GradientBoostingRegressor()
model=RandomForestRegressor()
model.fit(dftt_pp,dftt_label)
```

Out[17]:
```
RandomForestRegressor()
```

In [18]:
```
dftt_pp_lol=model.predict(dftt_pp)
dftt_pp_lol
```

Out[18]:
```
array([22.43 , 25.644, 16.492, 23.377, 23.472, 17.211, 24.547, 26.463,
       10.703, 32.362, 15.885, 18.366, 19.983, 34.224, 22.796, 13.96 ,
       18.515, 17.613,  7.009, 20.415, 19.019, 17.128, 18.217, 27.306,
       24.388, 24.702, 13.578, 21.026, 14.931, 28.438, 14.439, 18.008,
       11.305, 13.104, 20.275, 27.199, 23.368, 47.444, 33.336, 20.121,
       22.147, 35.529, 36.596,  9.465, 20.084, 12.501, 21.41 , 27.75 ,
       22.851,  7.582, 25.005, 22.937, 23.714, 31.023, 14.927, 37.882,
       21.576, 25.464, 33.561, 17.118, 19.801, 19.786, 28.038, 21.11 ,
       14.349, 31.547, 49.628, 14.865, 26.958, 34.695, 22.405, 36.468,
       23.315, 10.064, 20.599, 27.523, 25.679, 15.234, 16.358, 18.813,
       31.757,  8.523, 19.443, 18.912, 26.104, 23.363, 30.214, 25.098,
       29.259, 16.509, 23.041, 29.776, 28.039, 20.016, 21.124, 34.595,
       29.789, 19.926, 23.372, 17.039, 12.476, 46.059, 20.337, 44.178,
       16.931, 15.897, 19.85 , 21.87 , 27.502, 17.16 , 17.053,  7.714,
       14.927, 20.815, 11.292, 18.931, 13.842, 36.856, 24.377, 15.857,
       22.615,  8.33 ,  8.733, 19.745, 24.061, 14.15 , 15.96 , 24.223,
       12.744, 21.804, 23.015, 12.69 , 18.345, 12.101, 47.853, 17.927,
       23.9  , 23.542, 22.007, 49.41 , 15.84 , 21.659, 27.773, 19.098,
       32.835, 17.682, 21.138, 19.152, 20.523,  8.894, 23.9  , 16.151,
       36.221, 16.123, 26.822, 13.853, 18.214, 19.489, 24.121, 19.398,
       26.197, 24.708, 28.185, 19.441, 47.764, 15.974, 24.077, 48.166,
       26.248, 23.886, 20.981, 27.159, 44.871, 22.529, 22.01 , 23.065,
       14.063, 24.094,  8.864, 19.015, 41.594, 14.407, 20.958, 29.738,
       22.053, 20.545, 18.714, 27.364, 20.719, 31.031, 21.555, 30.517,
       21.636, 24.049, 16.036, 11.87 , 32.866, 31.222, 34.536, 23.148,
       17.775, 49.414, 19.585, 21.857, 28.846, 17.61 , 14.278, 22.709,
       22.386, 12.482, 26.391, 16.878, 22.848, 45.872, 21.948, 33.674,
       15.218, 27.845, 14.524, 10.718, 22.633, 19.544, 19.387, 18.267,
       19.278, 14.319, 10.571, 21.646, 19.762, 19.988, 17.758, 23.311,
       24.351, 16.972, 25.373, 34.807, 20.351, 10.09 , 13.86 , 20.734,
       13.594,  9.431, 31.127, 18.314, 21.333, 20.727, 22.711, 23.378,
       24.449, 30.623, 12.605, 20.999, 20.018, 21.085, 19.775, 16.313,
       48.186, 47.773, 19.599, 11.941, 14.443, 45.318, 25.63 ,  9.943,
       46.064, 14.285, 46.577, 23.621, 26.93 , 17.332, 35.82 , 22.069,
       19.167, 21.626, 22.565, 32.567, 14.409, 13.051, 20.112, 33.281,
       17.592, 22.52 , 17.615, 31.115, 36.163, 23.699, 21.292, 19.343,
       22.564, 46.898, 22.196, 11.163, 11.657, 13.653, 16.019, 20.589,
       15.192, 41.7  , 15.318, 47.741, 20.77 , 22.106, 45.819, 34.721,
       21.592,  8.09 ,  8.005, 23.561, 20.337, 20.154, 17.429, 27.842,
       19.28 , 21.35 , 23.786, 25.03 , 35.428, 29.294, 43.796, 21.153,
       17.614, 19.79 , 20.622, 17.071, 22.115, 47.93 , 20.252, 17.638,
       23.902, 17.003, 34.94 , 18.939, 19.343, 19.396, 24.188,  9.679,
       20.96 , 19.101, 17.777, 18.475, 18.419, 18.739, 12.648, 23.19 ,
       25.2  , 16.812, 18.596, 19.113, 15.227, 20.49 , 20.775, 14.519,
       23.755, 21.917, 19.46 , 22.187, 20.352, 30.209, 20.487, 22.101,
       17.955, 13.134, 13.895, 25.62 , 35.495, 47.412, 22.165, 18.59 ,
       23.479, 19.028, 13.747, 30.41 , 23.278, 46.601, 14.67 , 14.462,
       20.505,  8.644, 23.599, 19.453, 41.059, 28.973, 18.126, 13.595,
       13.173, 19.528, 39.907, 24.042, 33.948, 18.953, 24.38 ,  9.492,
        8.742, 16.122, 14.966, 26.174, 32.986,  6.406, 20.805, 16.213,
       13.468, 44.251, 14.39 , 23.007])
```

Validating the Algorithm

In [19]:
```
list(dftt_label.head(5))
```

Out[19]:
```
[21.9, 24.5, 16.7, 23.1, 23.0]
```

In [20]:
```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(dftt_label,dftt_pp_lol)
rmse=np.sqrt(mse)
rmse
```

Out[20]:
```
1.1977496670220713
```

Cross-Validation

In [21]:
```
from sklearn.model_selection import cross_val_score
score= cross_val_score(model,dftt_pp,dftt_label,scoring='neg_mean_squared_error',cv=10)
rmse2=np.sqrt(-score)
rmse2
```

Out[21]:
```
array([2.82153053, 2.92829375, 4.42278777, 2.43916311, 3.4476478 ,
       2.73790615, 4.56633161, 3.34982587, 3.28067127, 3.35713055])
```

Prediction using Test Set

In [22]:
```
final_test1=strat_test_set.drop(['MEDV'],axis=1)
final_test2=strat_test_set['MEDV']
```

In [23]:
```
final_test=my_pipe.fit_transform(final_test1)
test_prediction=model.predict(final_test)
test_prediction
```

Out[23]:
```
array([24.76 ,  9.767, 25.608, 20.983, 17.214, 14.321, 18.843, 14.692,
       32.161, 42.634, 18.869,  8.859, 24.729, 27.552, 20.387, 12.508,
       32.603, 14.737, 23.418, 17.218, 19.16 , 15.306, 16.267, 21.537,
       18.531, 31.095, 16.434, 32.219,  9.809, 34.459, 23.744, 20.953,
       22.969,  8.823, 19.969, 12.564, 42.174, 24.3  , 23.543, 44.254,
       23.222, 25.627, 19.634, 22.131, 15.671, 33.102, 44.966, 21.058,
       18.461, 21.885, 21.432, 14.581, 18.89 , 15.118, 25.222, 33.716,
       41.115, 28.855, 18.307, 21.093, 46.742, 10.633, 19.005, 22.923,
       14.463, 39.433, 19.158, 16.788, 17.916, 34.523, 25.516, 22.885,
       19.916, 23.214, 35.395, 13.093, 15.351, 20.243, 21.495, 21.462,
       22.538, 21.709, 14.077, 23.259, 21.397, 21.324, 14.409, 20.948,
       21.818, 23.625, 17.839, 26.918,  8.947, 25.743, 15.812, 30.41 ,
       19.631, 31.136, 13.803, 26.032, 18.536, 16.841])
```

In [24]:
```
from joblib import dump, load
dump(model, 'PricePrediction.joblib')
```

Out[24]:
```
['PricePrediction.joblib']
```

In [25]:
```
from joblib import dump, load
import numpy as np
model = load('PricePrediction.joblib')
features = np.array([[4.87141,  0.0, 18.10, 0, 0.614, 6.484, 93.6, 2.3053, 666, 20.2, 396.21, 18.68, 102.714374]])
model.predict(features)
```

Out[25]:
```
array([20.313])
```

In [ ]:
```
```