In [2]:
```
import pandas as pd
```

In [3]:
```
import numpy as np
```

In [4]:
```
df=pd.read_csv(r'C:\Users\Rutu\Documents\PDF\housingData.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN

In [6]:
```
df.tail()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	NaN
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	7.88

In [7]:
```
df.describe()
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.00
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.79
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.10
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.10
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.20
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.18
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

In [8]:
```
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     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 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    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

In [9]:
```
df.isnull()
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	False	False	False	False	False	False	False	False	False	False	False	False	True
502	False	False	False	False	False	False	False	False	False	False	False	False	False
503	False	False	False	False	False	False	False	False	False	False	False	False	False
504	False	False	False	False	False	False	False	False	False	False	False	False	False
505	False	False	False	False	False	False	True	False	False	False	False	False	False

506 rows × 14 columns

In [10]:
```
df.columns
```

Out[10]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

In [12]:
```
df.shape
```

Out[12]:
```
(506, 14)
```

In [13]:
```
df.dropna(inplace=True)
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
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 [15]:
```
df.value_counts()
```

Out[15]:
```
CRIM      ZN    INDUS  CHAS  NOX    RM     AGE    DIS     RAD  TAX  PTRATIO  B
LSTAT  MEDV
0.00632   18.0  2.31   0.0   0.538  6.575  65.2   4.0900  1    296  15.3     39
6.90   4.98   24.0     1
1.34284   0.0   19.58  0.0   0.605  6.066  100.0  1.7573  5    403  14.7     35
3.89   6.43   24.3     1
1.25179   0.0   8.14   0.0   0.538  5.570  98.1   3.7979  4    307  21.0     37
6.57   21.02  13.6     1
1.23247   0.0   8.14   0.0   0.538  6.142  91.7   3.9769  4    307  21.0     39
6.90   18.72  15.2     1
1.20742   0.0   19.58  0.0   0.605  5.875  94.6   2.4259  5    403  14.7     29
2.29   14.43  17.4     1

..
0.11460   20.0  6.96   0.0   0.464  6.538  58.7   3.9175  3    223  18.6     39
4.96   7.73   24.4     1
0.11432   0.0   8.56   0.0   0.520  6.781  71.3   2.8561  5    384  20.9     39
5.58   7.67   26.5     1
0.11132   0.0   27.74  0.0   0.609  5.983  83.5   2.1099  4    711  20.1     39
6.90   13.35  20.1     1
0.11069   0.0   13.89  1.0   0.550  5.951  93.8   2.8893  5    276  16.4     39
6.90   17.92  21.5     1
88.97620  0.0   18.10  0.0   0.671  6.968  91.9   1.4165  24   666  20.2     39
6.90   17.21  10.4     1
Length: 394, dtype: int64
```

In [16]:
```
df[['ZN']].value_counts()
```

Out[16]:
```
ZN
0.0      291
20.0      17
80.0      10
25.0      10
22.0       9
12.5       8
90.0       5
95.0       4
45.0       4
40.0       4
33.0       3
30.0       3
60.0       3
70.0       3
75.0       3
35.0       2
28.0       2
52.5       2
55.0       2
21.0       2
82.5       2
85.0       2
18.0       1
17.5       1
100.0      1
dtype: int64
```

In [17]:
```
df[['CHAS']].value_counts()
```

Out[17]:
```
CHAS
0.0     367
1.0      27
dtype: int64
```

In [18]:
```
df.dtypes
```

Out[18]:
```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX          int64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

```
In [19]: y=df['CRIM']
```

```
In [20]: y
```

```
Out[20]: 0      0.00632
         1      0.02731
         2      0.02729
         3      0.03237
         5      0.02985
                  ...
         499    0.17783
         500    0.22438
         502    0.04527
         503    0.06076
         504    0.10959
         Name: CRIM, Length: 394, dtype: float64
```

```
In [21]: x=df[['ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','B','LSTAT
```

```
In [22]: x
```

Out[22]:

	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
5	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...
499	0.0	9.69	0.0	0.585	5.569	73.5	2.3999	6	391	19.2	395.77	15.10	17.5
500	0.0	9.69	0.0	0.585	6.027	79.7	2.4982	6	391	19.2	396.90	14.33	16.8
502	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0

394 rows × 13 columns

```
In [23]: x.shape
```

```
Out[23]: (394, 13)
```

```
In [24]: from sklearn.model_selection import train_test_split
```

```
In [25]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.4,random_state=2
In [26]: x_train.shape,x_test.shape,y_train.shape,y_test.shape

Out[26]: ((236, 13), (158, 13), (236,), (158,))
```

```
In [27]: from sklearn.linear_model import LinearRegression
```

```
In [28]: lr=LinearRegression()
```

```
In [29]: lr.fit(x_train,y_train)

Out[29]: LinearRegression()
```

```
In [30]: y_pred=lr.predict(x_test)
```

```
In [31]: y_pred.shape

Out[31]: (158,)
```

In [32]:
```
y_pred
```

Out[32]:
```
array([12.935922  ,  1.03139699, -4.21746078, 15.37065881,  0.89075298,
        3.03311676, 16.22803816,  0.3028599 ,  4.30871359,  0.79750878,
        6.19982726,  1.25171726,  0.03758206,  1.90877592, -1.41151168,
       -0.45842888, -1.61532464,  1.29981509, -2.70302105,  1.08515213,
       17.83363067,  4.59686925,  2.1008869 ,  1.17613716, 14.48983654,
        5.03153761,  2.02889521, -0.16995535, 20.7697108 , -1.89013628,
        1.59905481, -3.82478187, -2.48892578, -2.3897628 , -0.45922083,
       -0.50957521, -2.01987301,  2.69480245, -1.43813057,  3.48991821,
        2.18396115,  0.78986788, 15.34920081,  7.71650902,  2.9751829 ,
       -0.54515732,  1.72443869, 16.73994458,  0.56656552, -1.28724361,
       18.04844372,  4.91251065, 16.99132212, -0.58645378, 15.54398917,
       -2.68280868,  0.53872057, 15.77861889,  1.00122542, -0.77916292,
       21.30193144, -0.64593524, -0.0303628 ,  3.22270199,  2.71050247,
       -0.30599732, -2.34637883,  1.10346922, -0.78161217,  1.69111522,
        2.47521546, -2.68191181,  2.31284667,  2.89030912,  4.20696855,
        0.08396164, 17.61903684, -7.52871885, -4.54305258, -1.33888227,
       -1.96025224,  1.80855902,  3.95622385, 17.93841206, -0.96922524,
        3.08395508, -2.27614313, -1.90571445,  1.26285226, 16.8538469 ,
       12.22256521,  0.42279442, 12.02868615, 13.96345377,  5.6734962 ,
       19.61189164, -3.83248688, 12.15255625, -5.03561045, -0.16179549,
       -2.11891663, -8.51705207,  2.50259354,  8.21403607,  2.67824565,
       -1.96774668,  0.72223693, 16.24830325, -3.9002718 , -0.36049838,
        0.21330248,  2.91480528, -3.51148999, -1.54047874,  2.75287094,
       17.5991266 ,  2.1164853 , -0.16043837, -1.77787688, 23.36861704,
       -4.38321192,  1.79112468, 14.9083135 ,  7.27267044, 21.59840775,
        0.64609847,  0.42677874,  0.3841648 , 17.25855024, 10.61349429,
        3.7032961 , 17.62947639, -3.42767243,  0.78833139, 15.82313715,
        0.42374552, 14.11204082, -0.19093817,  0.1038211 ,  1.42919613,
        0.1359542 , -0.29655227,  2.50152764,  1.36549247, 13.78581421,
       13.37325621, 16.95424315, -0.4363879 , -0.68717684,  2.33676091,
        2.74588406, 20.50932388, -0.74607058, -2.58931479, 13.83173767,
        1.43583166,  3.2355107 , -6.68796363])
```

In [33]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

In [34]:
```
mean_squared_error(y_test,y_pred)
```

Out[34]:
```
23.57799042915025
```

In [35]:
```
mean_absolute_error(y_test,y_pred)
```

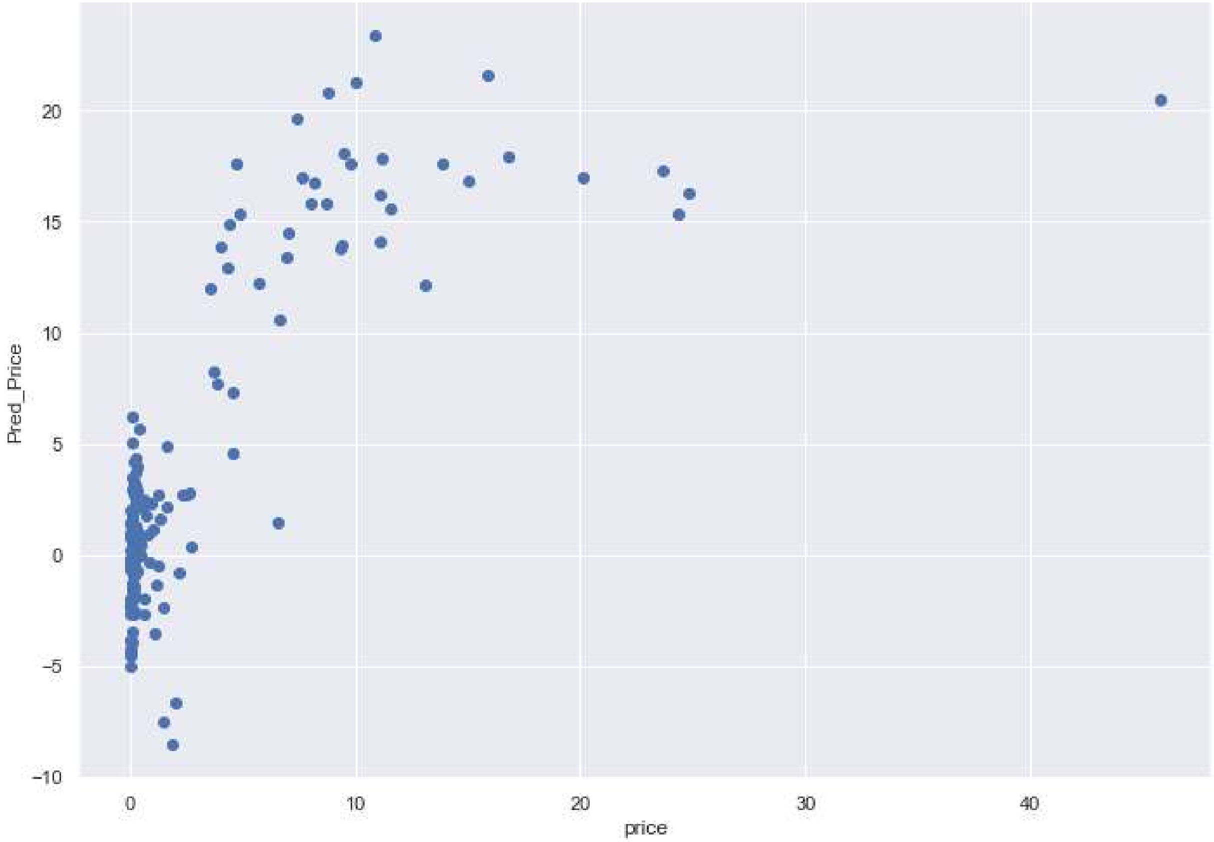
Out[35]:
```
3.330071652883863
```

In [36]:
```
r2_score(y_test,y_pred)
```

Out[36]:
```
0.3607529980840608
```

In [46]:

```
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.xlabel("price")
plt.ylabel("Pred_Price")
plt.show()
```

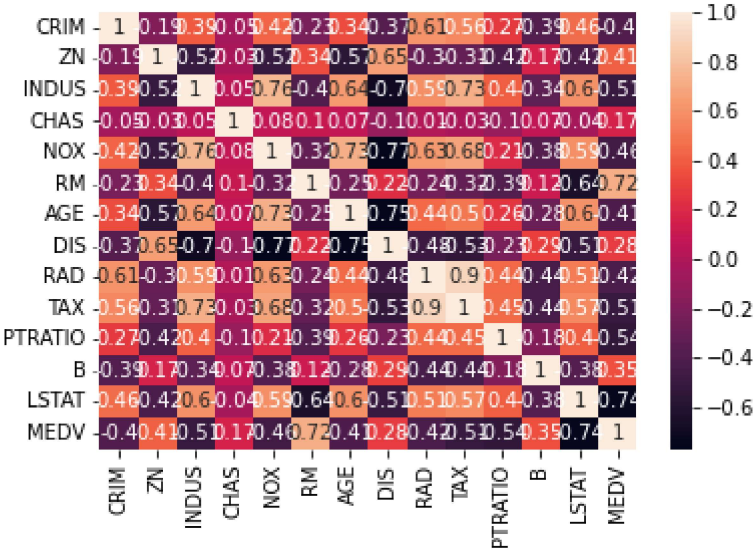


In [38]:

```
import seaborn as sns
```

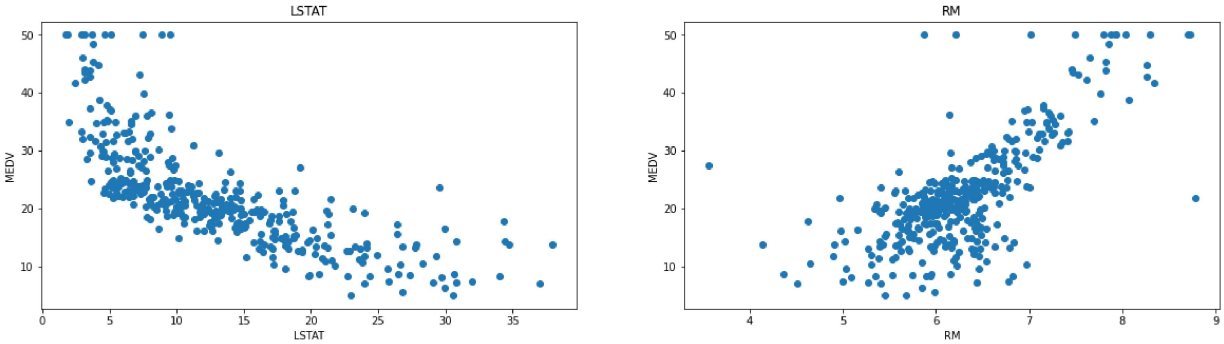
In [40]:
```
correlation_matrix = df.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
```

Out[40]:
```
<AxesSubplot:>
```



In [43]:

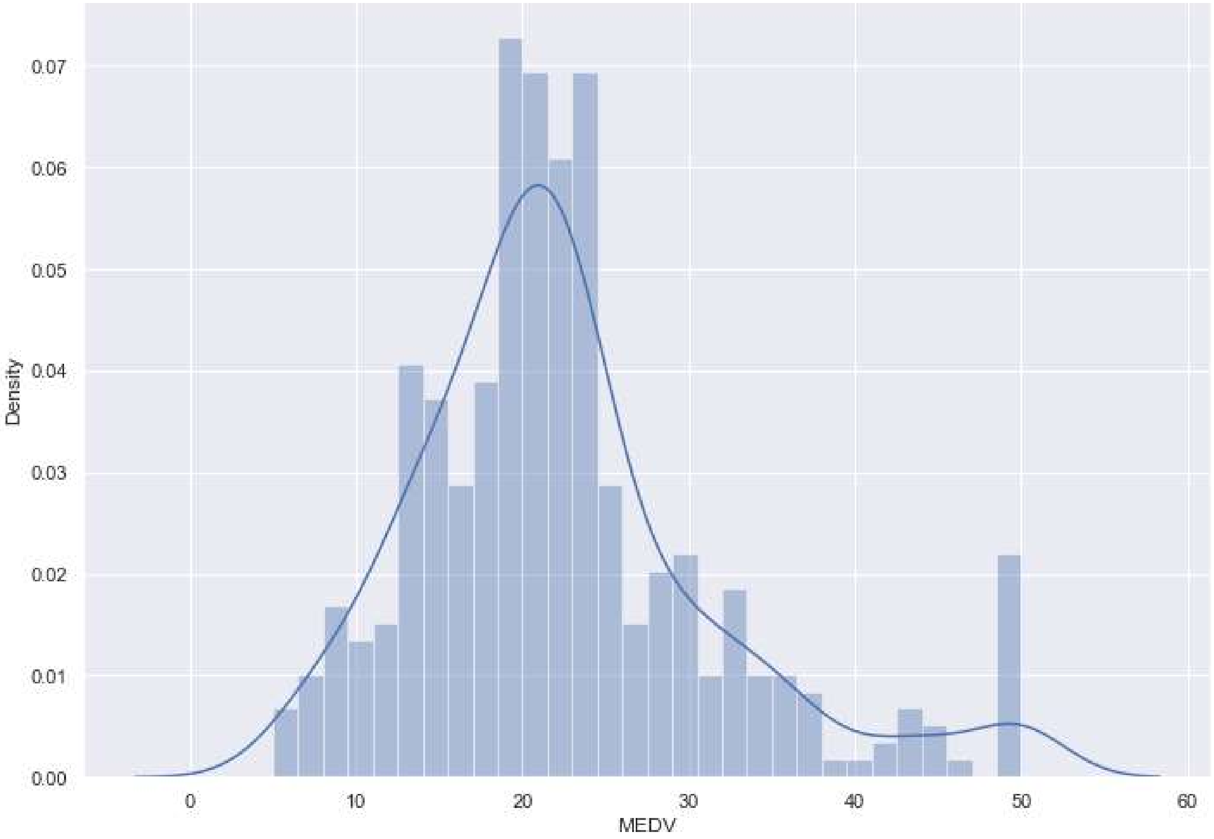
```
plt.figure(figsize=(20, 5))

features = ['LSTAT', 'RM']
target = df['MEDV']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = df[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



In [44]:

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(df['MEDV'], bins=30)
plt.show()
```

```
C:\Users\Rutu\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Future
Warning: `distplot` is a deprecated function and will be removed in a future ve
rsion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```


Density
MEDV