In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df = pd.read_csv("HousingData.csv")
```

In [3]:

```
df
```

Out[3]:

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

506 rows × 14 columns

In [4]:

```
df.head()
```

Out[4]:

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

In [5]:

```
df.tail()
```

Out[5]:

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

In [6]:

```
df.shape
```

Out[6]:

```
(506, 14)
```

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
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

In [8]:

```
df.dtypes
```

Out[8]:

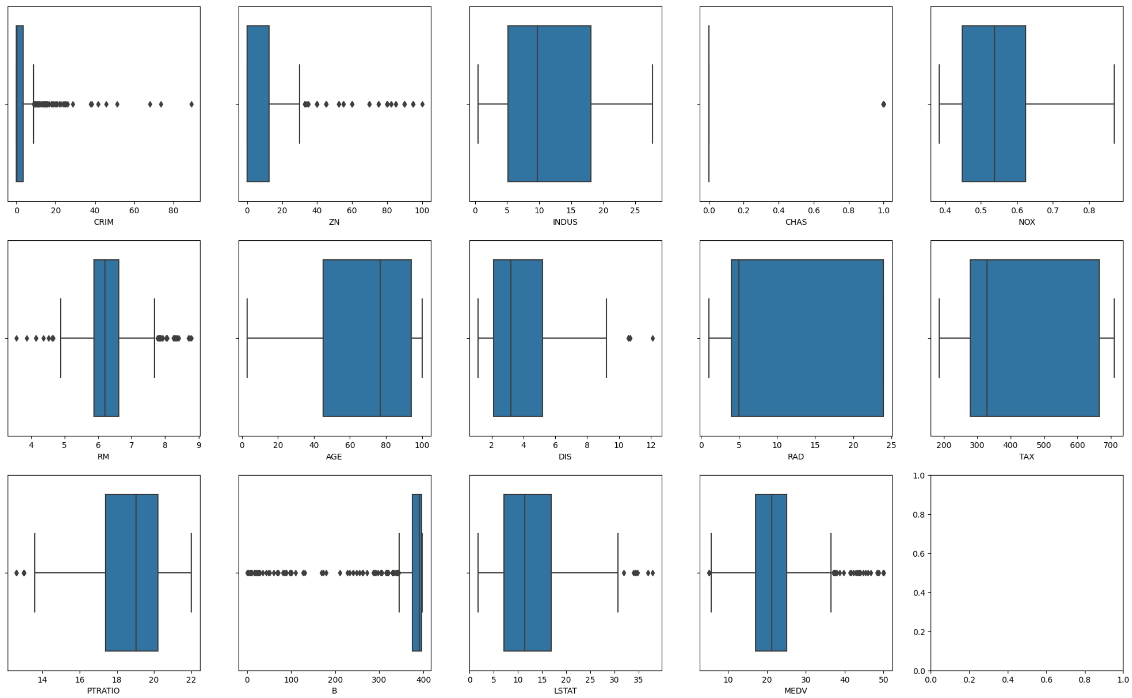
```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX          int64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

In [9]:

```
import matplotlib.pyplot as plt
```

In [10]:

```
fig , axs = plt.subplots(ncols = 5,nrows = 3,figsize =(25,15))
index = 0
axs = axs.flatten()
for k,v in df.items():
    sns.boxplot(v,ax=axs[index])
    index = index + 1
```



In [11]:

```
for k,v in df.items():
    q1 = v.quantile(0.25)
    q3 = v.quantile(0.75)
    IQR = q3-q1
    v_col = v[(v<=q1-1.5*IQR)| (v>=q3 +1.5*IQR)]
    perc = np.shape(v_col)[0] * 100.0 / np.shape(df)[0]
    print("Column %s has outliers :-  %.2f%%" %(k,perc))
```

```
Column CRIM has outliers :-  12.85%
Column ZN has outliers :-  12.45%
Column INDUS has outliers :-  0.00%
Column CHAS has outliers :-  96.05%
Column NOX has outliers :-  0.00%
Column RM has outliers :-  5.93%
Column AGE has outliers :-  0.00%
Column DIS has outliers :-  0.99%
Column RAD has outliers :-  0.00%
Column TAX has outliers :-  0.00%
Column PTRATIO has outliers :-  2.96%
Column B has outliers :-  15.22%
Column LSTAT has outliers :-  1.38%
Column MEDV has outliers :-  7.91%
```

In [12]:

```
df = df[~(df['MEDV'] >= 50.0)]
```

In [13]:

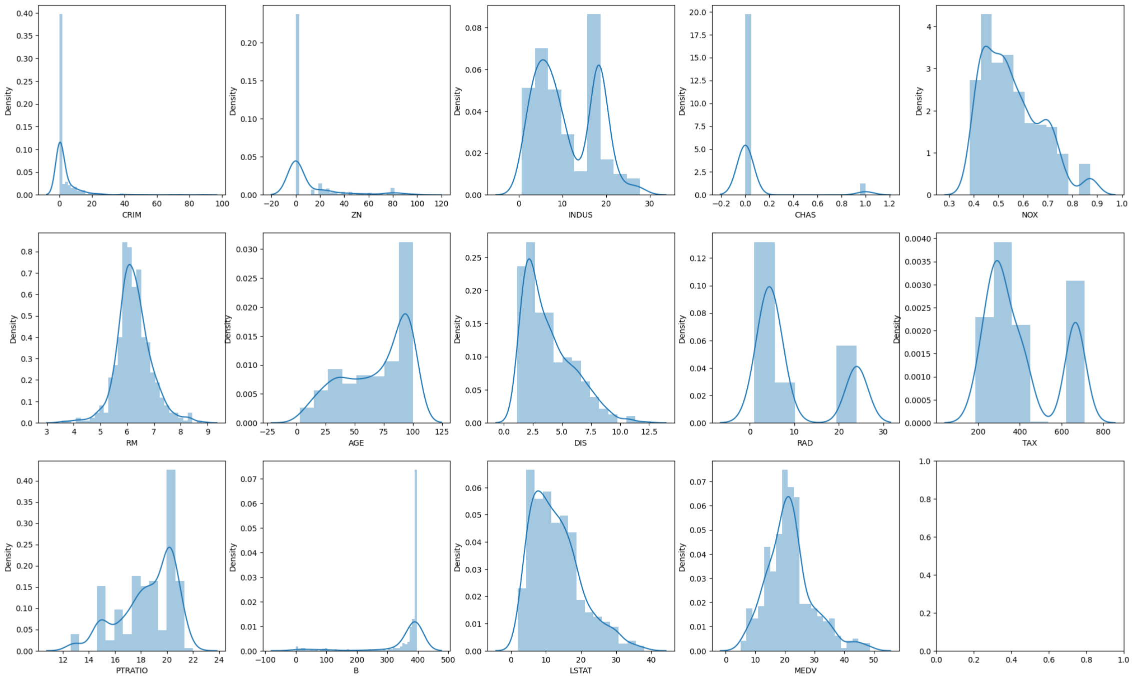
```
df.shape
```

Out[13]:

```
(490, 14)
```

In [14]:

```
fig,axs = plt.subplots(nrows = 3,ncols = 5,figsize=(25,15))
index = 0
axs = axs.flatten()
for k,v in df.items():
    sns.distplot(v,ax = axs[index])
    index = index+1
```



The histogram also shows that columns CRIM, ZN, B has highly skewed distributions.

Also MEDV looks to have a normal distribution (the predictions) and other colums seem to have normal or bimodel ditribution of data except CHAS (which is a discrete variable).

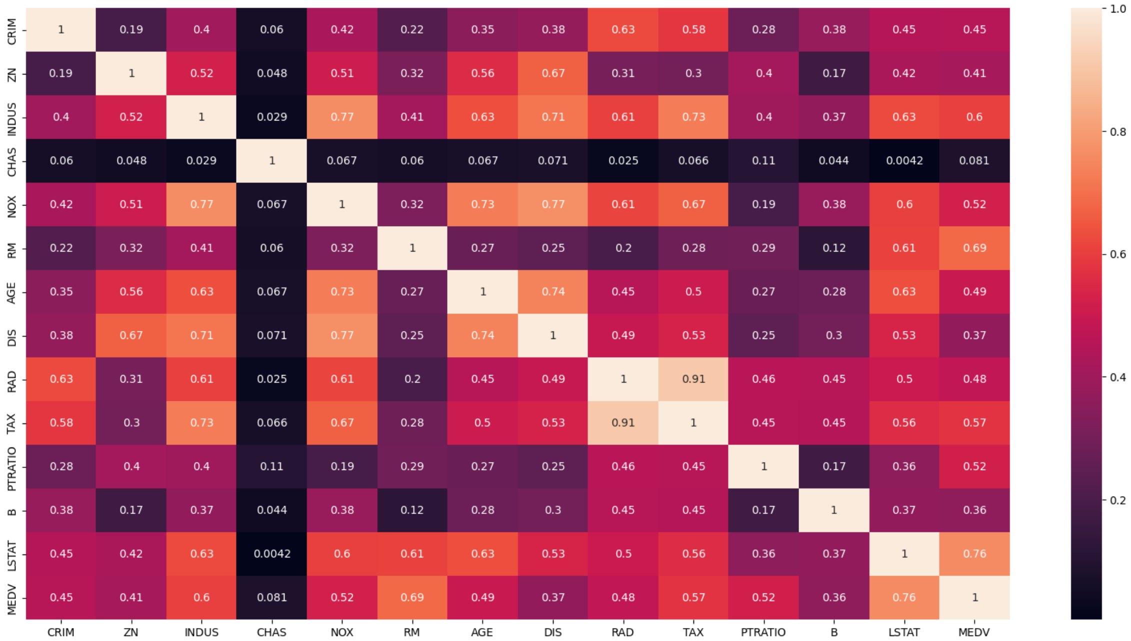
Now let's plot the pairwise correlation on data

In [15]:

```
plt.figure(figsize = (20,10))
sns.heatmap(df.corr().abs(),annot = True)
```

Out[15]:

```
<Axes: >
```



In [16]:

```
#From correlation matrix, we see TAX and RAD are highly correlated features.
#The columns LSTAT, INDUS, RM, TAX, NOX, PTRAIO has a correlation score above 0.5 with M
#indication of using as predictors. Let's plot these columns against MEDV.
```

In [ ]:

```

```

In [17]:

```
df.isna().sum()
```

Out[17]:

```
CRIM       19
ZN         19
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        18
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

In [19]:

```
df = df.fillna(df.mean())
```

In [20]:

```
df.isna().sum()
```

Out[20]:

```
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 [ ]:

```
#Traing the dataset and testing it
```

In [21]:

```
X = df[['LSTAT','RM']]
Y = df['MEDV']
```

In [22]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2,random_state = 45)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(392, 2)
(98, 2)
(392,)
(98,)
```

In [23]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

model = LinearRegression()
model.fit(X_train,Y_train)
```

Out[23]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:

```
# model evaluation for training set
```

In [25]:

```
from sklearn.metrics import r2_score
Y_train_predict = model.predict(X_train)
rmse = np.sqrt(mean_squared_error(Y_train,Y_train_predict))
r2 = r2_score(Y_train,Y_train_predict)
print("Model performance for Training set")
print(f"rmse score is = {rmse}")
print(f"r2 score is = {r2}")
```

```
Model performance for Training set
rmse score is = 4.8078921735147
r2 score is = 0.6201774308171788
```

In [26]:

```
Y_test_predict = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(Y_test,Y_test_predict))
r2 = r2_score(Y_test,Y_test_predict)
print("Model performance for testing set is ")
print(f"rmse score is = {rmse}")
print(f"r2 score is = {r2}")
```

```
Model performance for testing set is
rmse score is = 4.386171868330308
r2 score is = 0.704574214564875
```

In [ ]:

```
```

In [42]:

```
x = model.predict([[4.98,6.575]])
print(x)
print(Y_test_predict[0])
```

```
[27.63583413]
31.40957350799178
```

In [ ]:

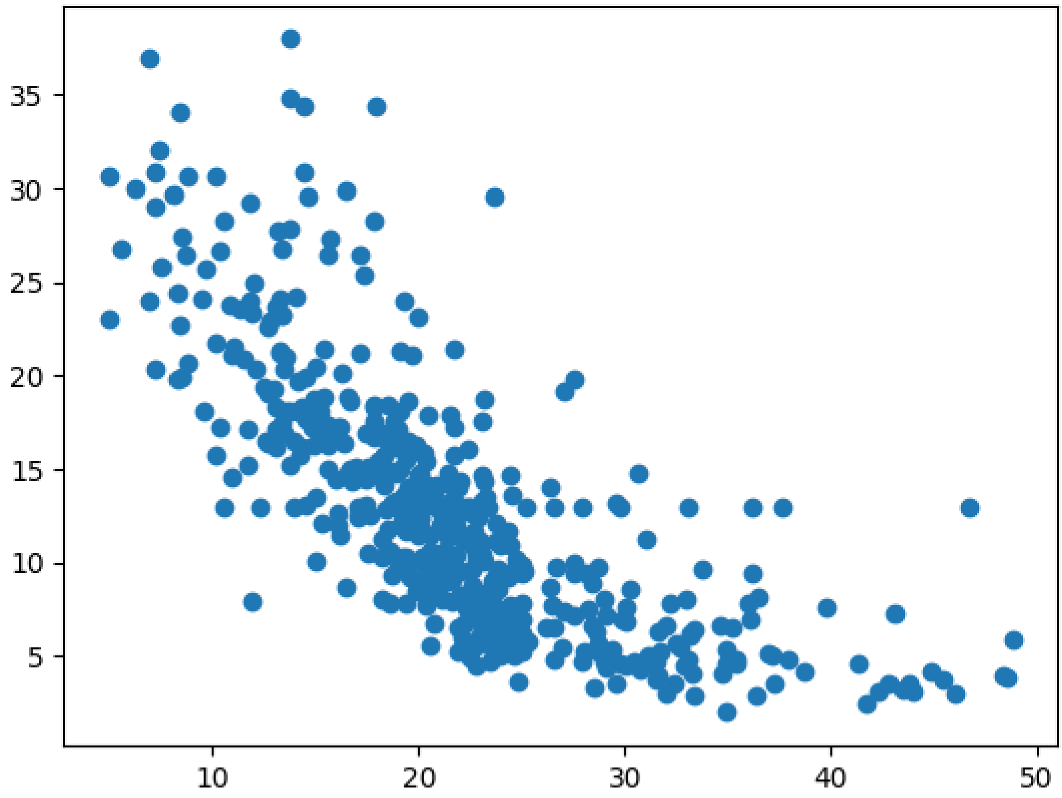
```
```

In [44]:

```
plt.scatter(x = df['MEDV'],y=df['LSTAT'])
```

Out[44]:

```
<matplotlib.collections.PathCollection at 0x20f390bb460>
```

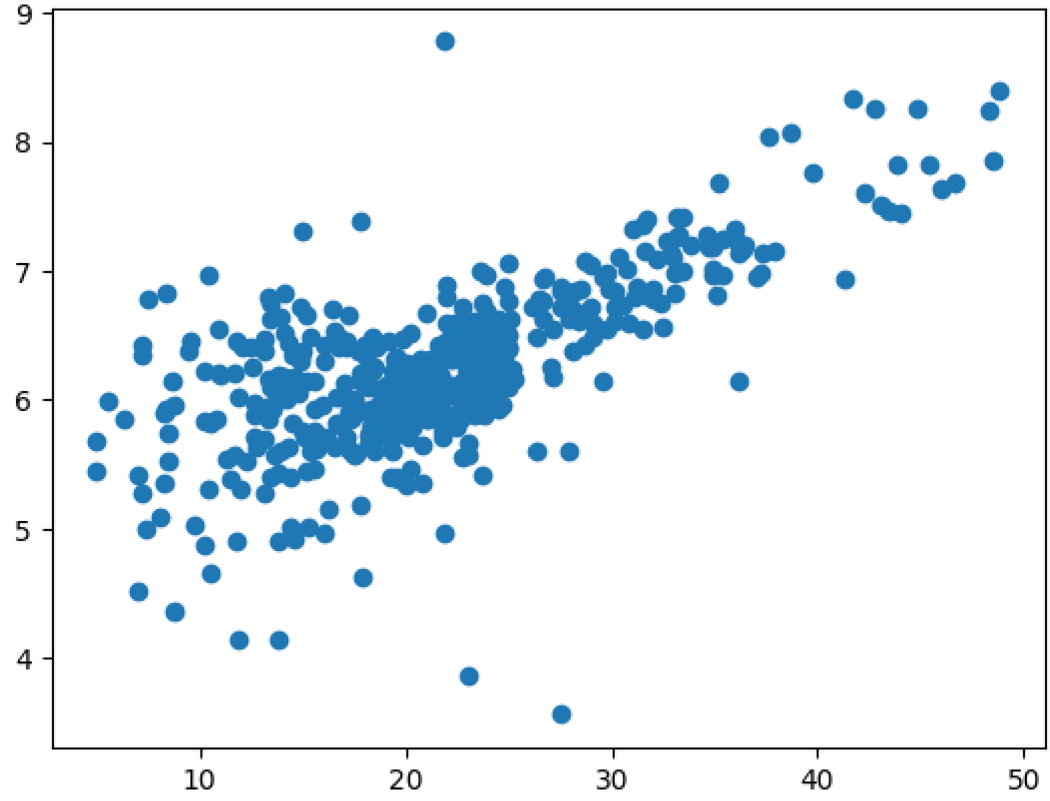


In [45]:

```
plt.scatter(x = df['MEDV'],y=df['RM'])
```

Out[45]:

```
<matplotlib.collections.PathCollection at 0x20f38f4fdf0>
```



In [ ]: