BOSTON HOUSE PRICE DATASET

Dataset Information

Boston House Price Dataset was collected in 1987 and has 506 entries with 14 attributes.

Boston House Price Dataset Attributes Information :

- CRIM: Per capita crime rate by town
- ZN: Proportion of residential land zoned for lots over 25,000 sq. ft
- INDUS: Proportion of non-retail business acres per town
- CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX: Nitric oxide concentration (parts per 10 million)
- RM: Average number of rooms per dwelling
- AGE: Proportion of owner-occupied units built prior to 1940
- DIS: Weighted distances to five Boston employment centers
- RAD: Index of accessibility to radial highways
- PTRATIO: Pupil-teacher ratio by town
- B: $1000(Bk — 0.63)^2$, where Bk is the proportion of [people of African American descent] by town
- LSTAT: Percentage of lower status of the population
- MEDV: Median value of owner-occupied homes in $1000s

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

In [2]:
```
df=pd.read_csv("Data (1).csv")
```

In [3]:
```
df.head()
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	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	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	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	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

Display Attributes Name

In [4]:
```
# display column name
print(df.columns.values)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT' 'MEDV']
```

Display Shape(Row and columns)

In [5]:
```
print(f"Display Number of Row and Columns : {df.shape}")
```

```
Display Number of Row and Columns : (506, 14)
```

Display Datatypes information

In [6]:

```
print(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     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
None
```

Display Statistical information

In [7]:

```
df.describe()
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.79
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.10
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.10
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.20
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.18
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

Preprocessing the Dataset

check for null values

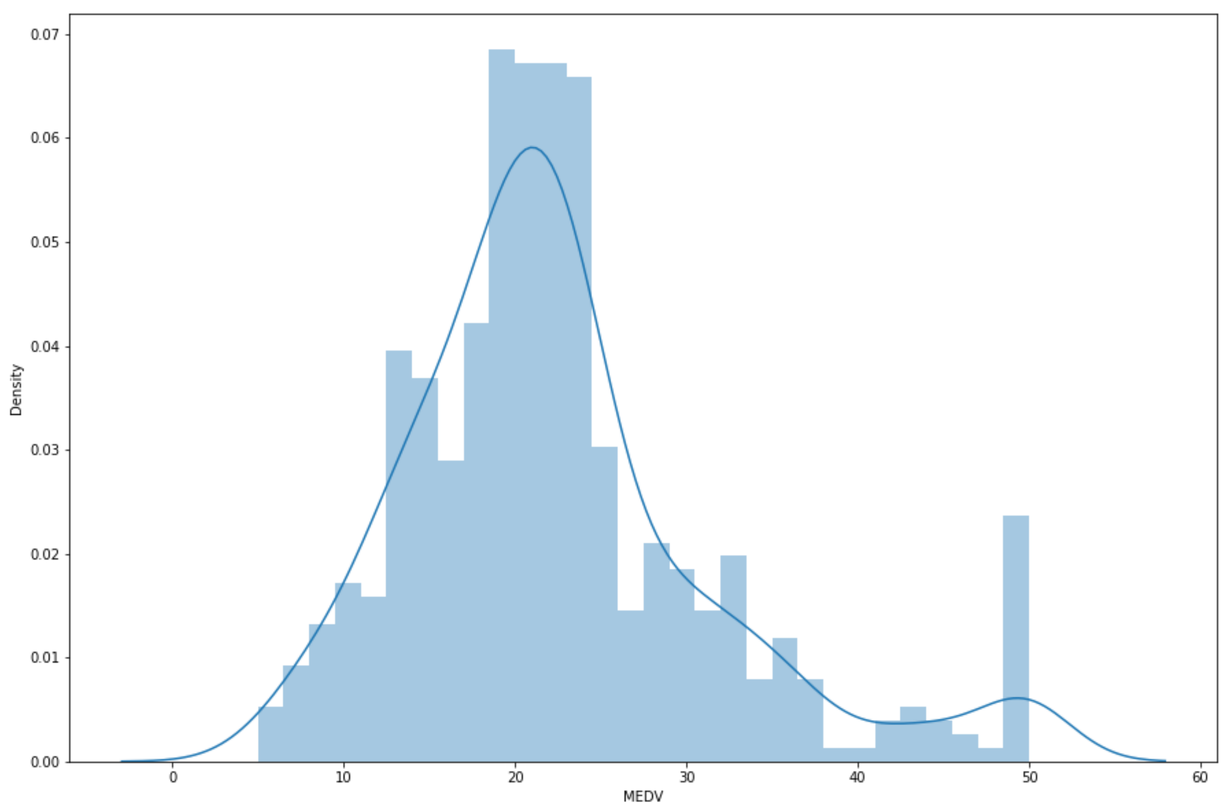
In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

Exploratory Data Analysis

Let's first plot the distribution of the target variable MEDV. We will use the distplot function from the seaborn library.

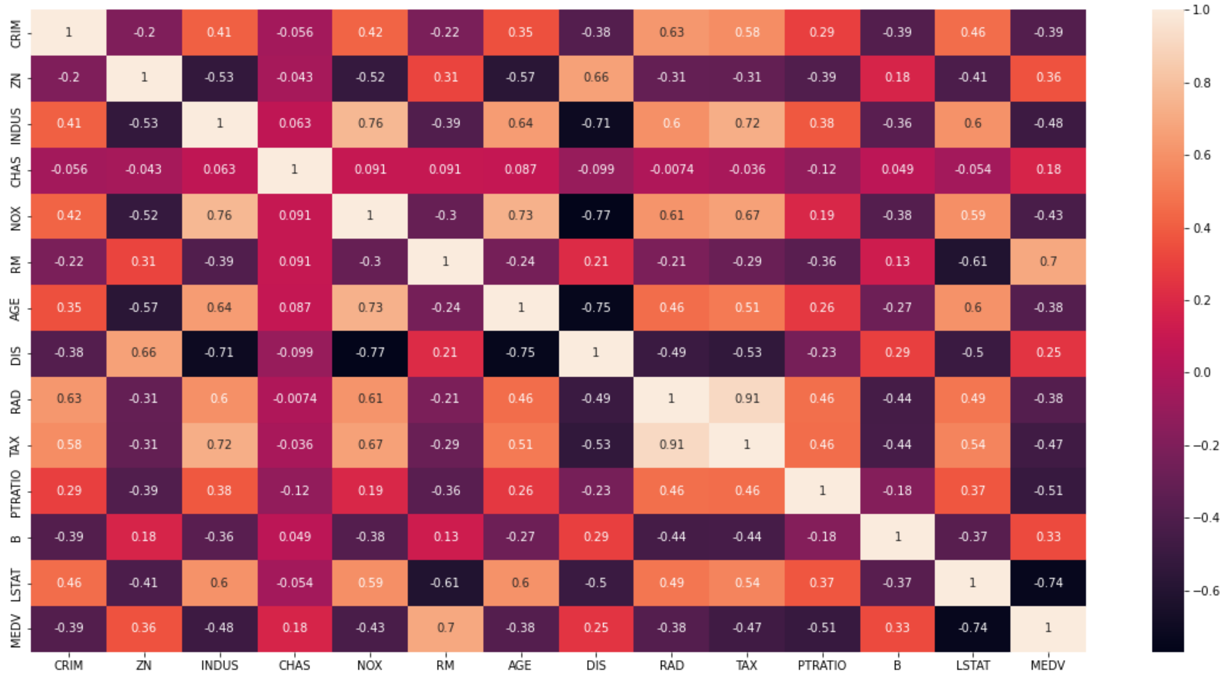
In [9]:
```
plt.figure(figsize=(15,10))
sns.distplot(df['MEDV'], bins=30)
plt.show()
```



We see that the values of MEDV are distributed normally with few outliers.

Next, we create a correlation matrix that measures the linear relationships between the variables.

In [10]:
```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(), annot=True)
plt.show()
```



The correlation coefficient ranges from -1 to 1.

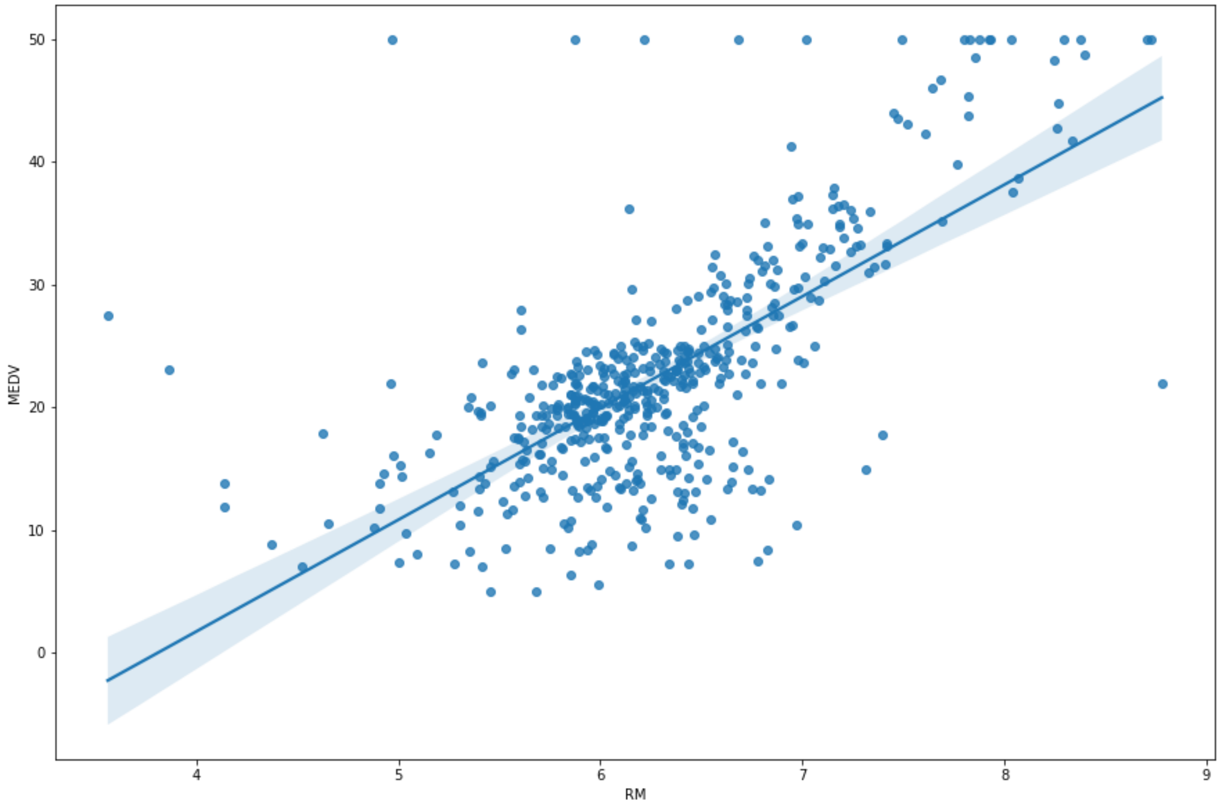
- If the value is close to 1, it means that there is a strong positive correlation between the two variables.
- When it is close to -1, the variables have a strong negative correlation.

Observations:

- By looking at the correlation matrix we can see that RM has a strong positive correlation with MEDV (0.7) where as LSTAT has a high negative correlation with MEDV(-0.74).
- An important point in selecting features for a linear regression model is to check for multi-co-linearity. The features RAD, TAX have a correlation of 0.91. These feature pairs are strongly correlated to each other. We should not select both these features together for training the model. Same goes for the features DIS and AGE which have a correlation of -0.75.

Based on the above observations we will RM and LSTAT as our features. Using a scatter plot let's see how these features vary with MEDV.

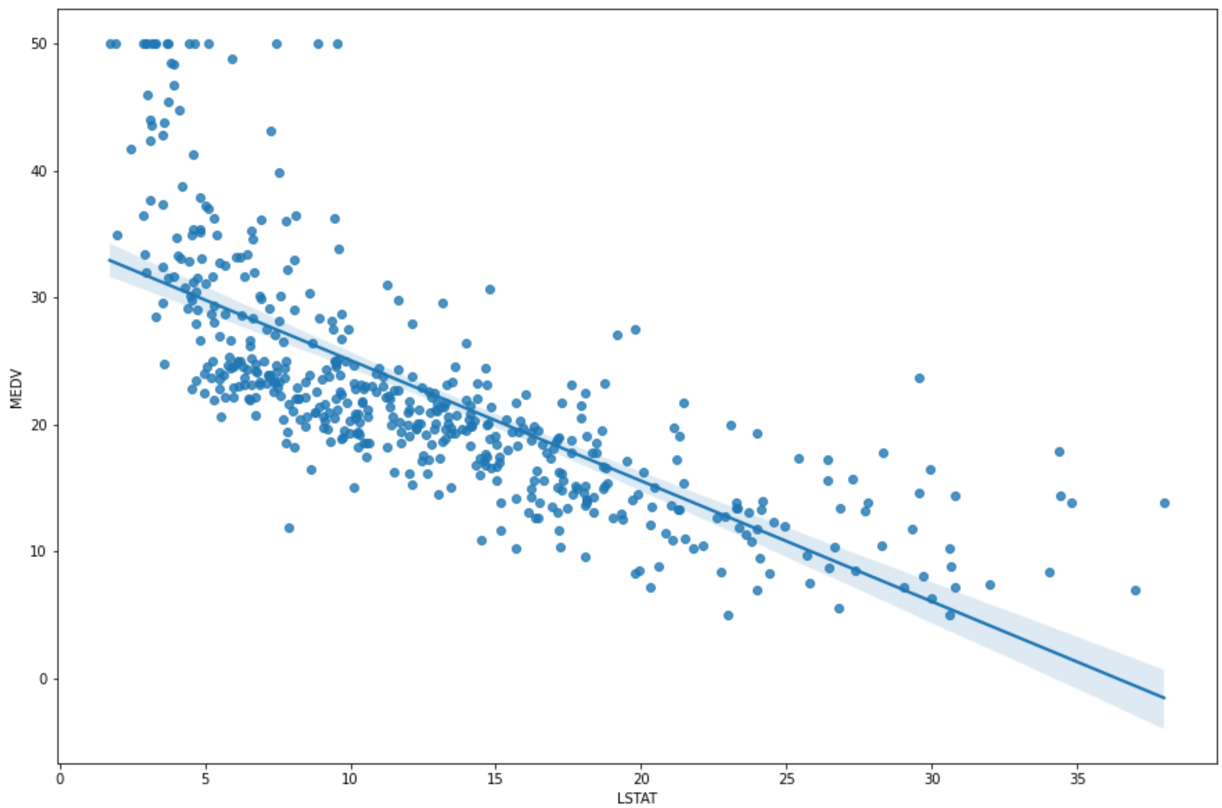
In [11]:
```
# Regression Plot
plt.figure(figsize=(15,10))
sns.regplot(y=df['MEDV'], x=df['RM'])
plt.show()
```



Observation:

- In Regression Plot we can see That the price of house increases then the RM also increases.

In [12]:
```
plt.figure(figsize=(15,10))
sns.regplot(y=df['MEDV'], x=df['LSTAT'])
plt.show()
```



Observation:

- In Regression Plot we can see That if the price of house decreases then the LSTAT increases.

Preparing the data for training the model

In [13]:
```
X= df.drop(columns=['MEDV'],axis=1)
y=df['MEDV']
```

Model Training

In [14]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30, random_st
```

Linear Regression

- Import library for Linear Regression

In [15]:
```
from sklearn import linear_model
# Create a Linear regression class
reg=linear_model.LinearRegression()
```

In [16]:
```
#Train the model using the training sets
reg.fit(X_train, y_train)
```

Out[16]:
```
LinearRegression()
```

In [17]:
```
# Accuracy
print(f"Accuracy Linear Regressor : {reg.score(X_test, y_test)}")
```

```
Accuracy Linear Regressor : 0.711226005748491
```

Predicted Result on Sample values Linear Regression

In [18]:
```
y_predict=reg.predict([[0.06905,0.0,2.18, 0, 0.458,7.147,54.2,6.0622, 3, 222, 18.
```

In [19]:
```
print(f"Result : {y_predict}")
```

```
Result : [28.20837173]
```

In [20]:
```
df['MEDV'][4]
```

Out[20]:
```
36.2
```

Random Forest

In [21]:
```
# Import Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
```

In [22]:
```
reg1 = RandomForestRegressor()
```

In [23]:
```
reg1.fit(X_train, y_train)
```

Out[23]:
```
RandomForestRegressor()
```

In [24]:
```
print(f"Accuracy RandomForest Regressor :{reg1.score(X_test, y_test)}")
```

```
Accuracy RandomForest Regressor :0.8777351623434168
```

Predicted Result on Sample values

In [25]:
```
y_predict1=reg1.predict([[0.06905,0.0,2.18, 0, 0.458,7.147,54.2,6.0622, 3, 222, 1
```

In [26]:
```
print(f"Result :{y_predict1}")
```

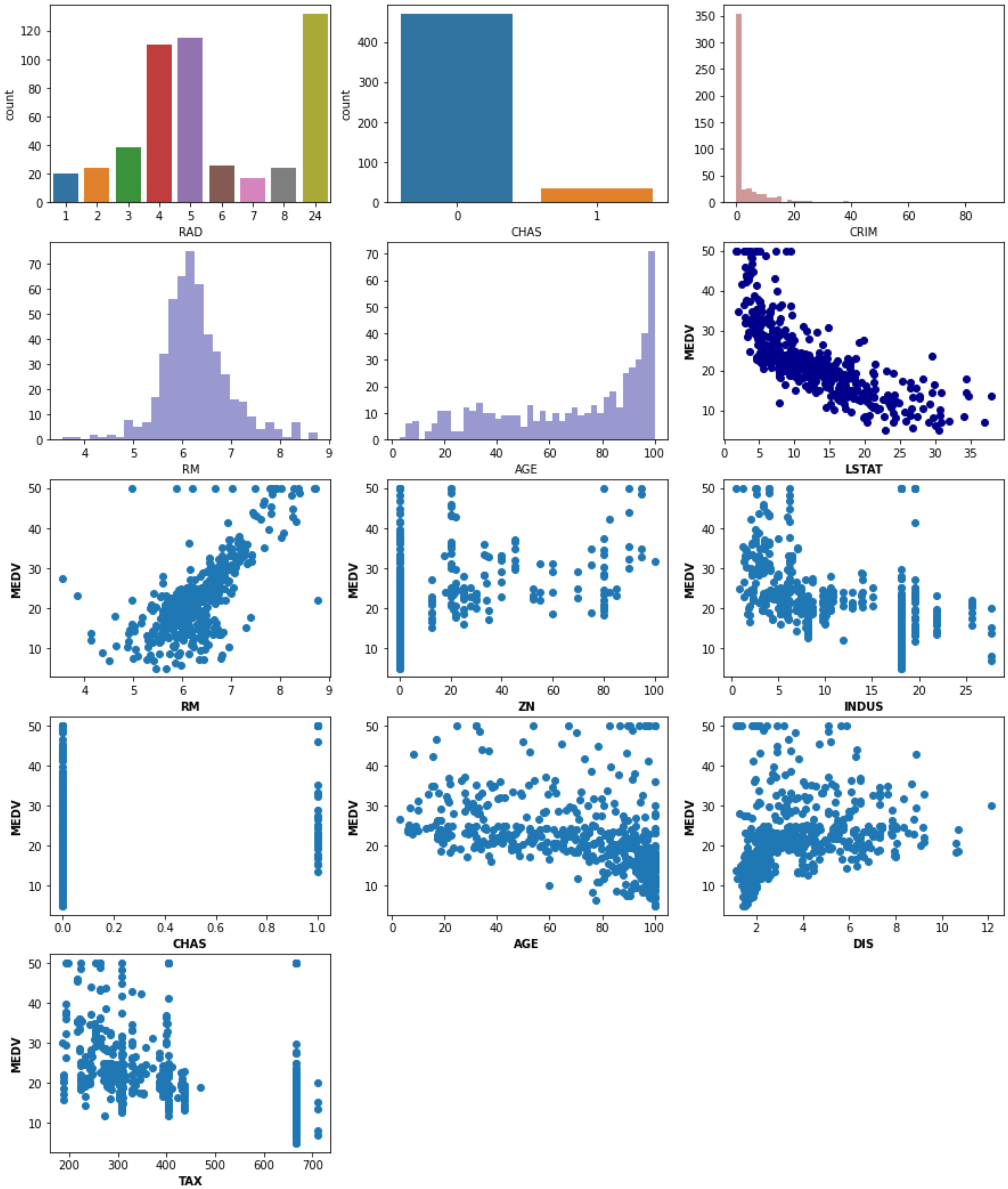
```
Result :[35.161]
```

In [27]:
```
df['MEDV'][4]
```

Out[27]:
```
36.2
```

DASHBOARD

In [30]:
```
plt.figure(figsize=[15,30])
plt.subplot(8, 3, 1)
sns.countplot(x='RAD',data=df)
plt.subplot(8, 3, 2)
sns.countplot(x='CHAS',data=df)
plt.subplot(8, 3, 3)
sns.distplot(df['CRIM'], kde=False, color='darkred')
plt.subplot(8, 3, 4)
sns.distplot(df['RM'], kde=False, color='darkblue')
plt.subplot(8, 3, 5)
sns.distplot(df['AGE'], kde=False, color='darkblue',bins=40)

plt.subplot(8, 3, 6)
plt.scatter(y=df['MEDV'], x=df['LSTAT'], c='darkblue')
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('LSTAT', fontsize=10.0, fontweight='bold')

plt.subplot(8, 3, 7)
plt.scatter(y=df['MEDV'], x=df['RM'])
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('RM', fontsize=10.0, fontweight='bold')

plt.subplot(8, 3, 8)
plt.scatter(y=df['MEDV'], x=df['ZN'])
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('ZN', fontsize=10.0, fontweight='bold')

plt.subplot(8, 3, 9)
plt.scatter(y=df['MEDV'], x=df['INDUS'])
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('INDUS', fontsize=10.0, fontweight='bold')

plt.subplot(8, 3, 10)
plt.scatter(y=df['MEDV'], x=df['CHAS'])
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('CHAS', fontsize=10.0, fontweight='bold')

plt.subplot(8, 3, 11)
plt.scatter(y=df['MEDV'], x=df['AGE'])
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('AGE', fontsize=10.0, fontweight='bold')

plt.subplot(8, 3, 12)
plt.scatter(y=df['MEDV'], x=df['DIS'])
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('DIS', fontsize=10.0, fontweight='bold')

plt.subplot(8, 3, 13)
plt.scatter(y=df['MEDV'], x=df['TAX'] )
plt.ylabel('MEDV', fontsize=10.0, fontweight='bold')
plt.xlabel('TAX', fontsize=10.0, fontweight='bold')

plt.show()
```


count
RAD
count
CHAS
CRIM
RM
AGE
MEDV
LSTAT
MEDV
RM
MEDV
ZN
MEDV
INDUS
MEDV
CHAS
MEDV
AGE
MEDV
DIS
MEDV
TAX

Observation:

- In first plot maximum RAD count is 24.
- In second plot maximum RAD count for 0.
- In third plot maximum CRIME Rate is 0.
- In fourth plot maximum number of Room is more than 6 in a particular house.
- In fifth plot maximum Age of house is 100.
- The prices increase as the value of RM increases.
- The prices tend to decrease as the value of LSTAT increase.
- The prices increase as the value of ZN increases.
- The prices tend to decrease as the value of INDUS increase.
- The prices increase as the value of CHAS increases.
- The prices tend to decrease as the AGE of House increase.
- The prices increase as the value of DIS increases.
- The prices tend to decrease as the value of TAX increase.

In [ ]: