EDA on Boston House Price Prediction

In this project we are going to use Machine Learning to predict the house prices of city named Boston in US.

The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970.

There are several features given for a house and we have to predicts its value as accurate as possible.

0. Overview

Below is the overview of the whole project, what all things we will be doing, step wise.

- 1. Importing Libraries
- 2. Exploring Dataset
    - 2.1. We will be importing the dataset using Pandas library.
    - 2.2. Finding variables which are useful for prediction.
- 3. Univariate and Multivariate Analysis

1. Importing Libraries

First we are importing all the important libraries we are going to use in this project and if we need any other library, we will import it at that time only.

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings #to remove warning from the notebook
warnings.filterwarnings(action='ignore')
```

```
df=pd.read_csv('/content/housing.csv')
df.head()
```

	0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30 396.90 4.98 24.00
0	0.02731 0.00 7.070 0 0.4690 6.4210 78...
1	0.02729 0.00 7.070 0 0.4690 7.1850 61...
2	0.03237 0.00 2.180 0 0.4580 6.9980 45...
3	0.06905 0.00 2.180 0 0.4580 7.1470 54...

2. Exploring Dataset

Boston House Price dataset has 14 features and their description is given as follows:

- 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 oxides 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 centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per dollar 10,000.
- PTRATIO pupil-teacher ratio by town
- B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in $1000's

Here main thing to notice is that MEDV is the outcome variable which we need to predict and all other variables are predictor variables.

2.1 Loading Dataset

Here we are going to import our **Boston House Price** dataset and will see how it looks o_o

```
#loading dataset
name= ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO',
df = pd.read_csv('housing.csv',delim_whitespace=True,names=name)
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	

```
#shape of our dataset
df.shape
```

```
(506, 14)
```

This data set has 14 features and 506 rows i.e. details of 506 houses.

```
#information about the data
df.dtypes
```

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

We can see that all features in the dataset are numeric type either float or int. There is no categorical variable.

```
#checking for missing data
df.isnull()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	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
2	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
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
502	False	False	False	False	False	False	False	False	False	False	False	False

We noticed that there are *No Missing* values in the dataset.

```
#total sum of missing values in each feature
df.isnull().sum()
```

```
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
```

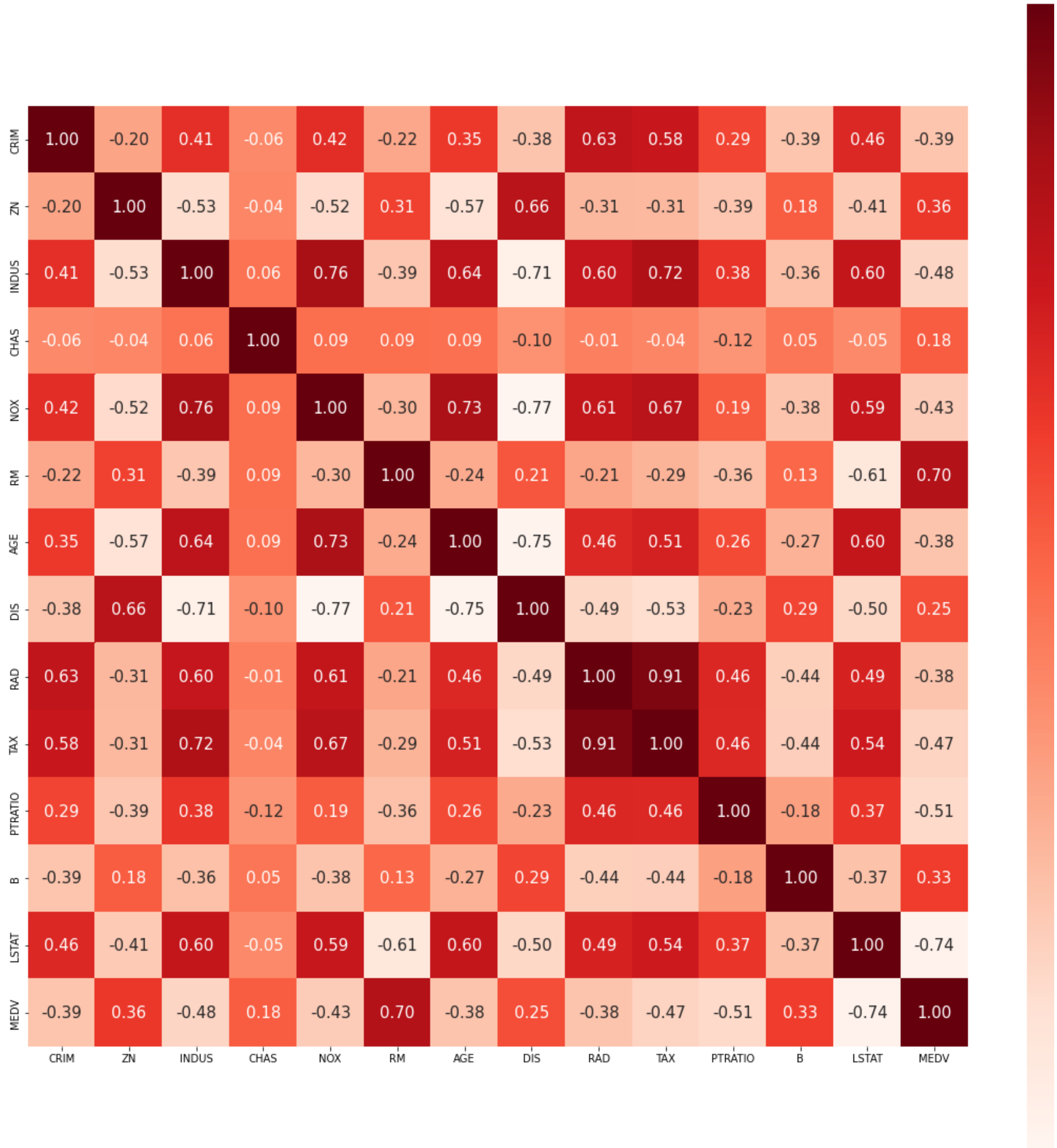
2.2 Finding variables which are useful for prediction

```
corr = df.corr()
corr.shape
```

```
(14, 14)
```

```
#print the correlation matrix
plt.figure(figsize=(20,20))
sns.heatmap(corr, cbar=True, square= True, fmt='.2f', annot=True, annot_kws={'size':15}, c
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7952715880>
```



Observing the heatmap, features showing correlation above 0.5 are strongly correlated; and features showing correlation below -0.5 are weakly correlated. Features between -0.5 and 0.5 may show multicollinearity.

```
#print the statistical report of the dataset
df.describe()
```

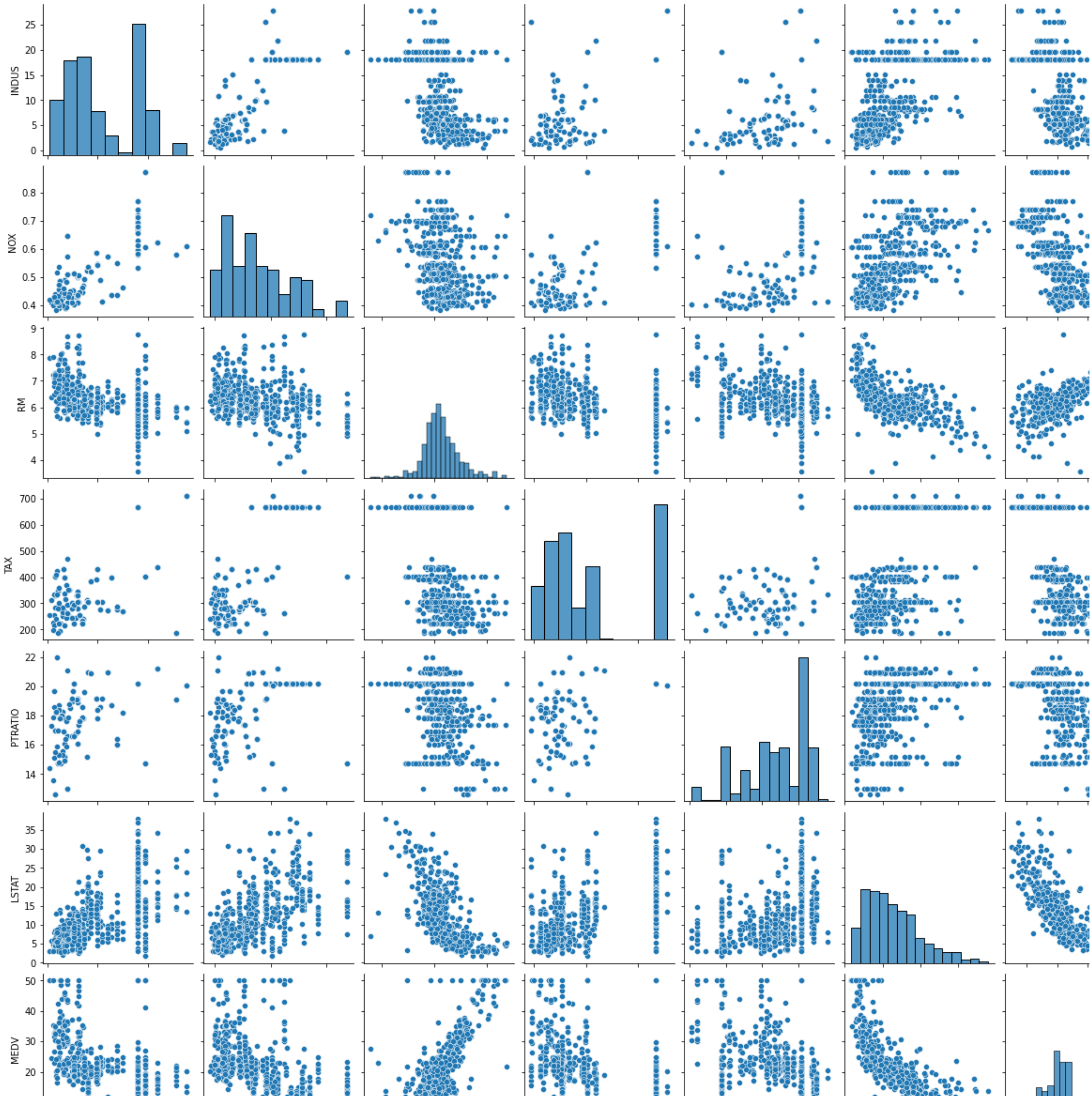
	CRIM	ZN	INDUS	CHAS	NOX	RM	A(
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.5749(
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.1488(
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.9000(
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.0250(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.5000(
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.0750(
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.0000(

```
#since some of these features shows quite good and very good correlation with our predicti
df1 = df[['INDUS', 'NOX', 'RM', 'TAX', 'PTRATIO', 'LSTAT', 'MEDV']]
df1.head()
```

	INDUS	NOX	RM	TAX	PTRATIO	LSTAT	MEDV
0	2.31	0.538	6.575	296.0	15.3	4.98	24.0
1	7.07	0.469	6.421	242.0	17.8	9.14	21.6
2	7.07	0.469	7.185	242.0	17.8	4.03	34.7
3	2.18	0.458	6.998	222.0	18.7	2.94	33.4
4	2.18	0.458	7.147	222.0	18.7	5.33	36.2

```
#generate the pairplot and write the inferences
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.pairplot(df1)
```

```
<seaborn.axisgrid.PairGrid at 0x7f794f4a1e20>
```



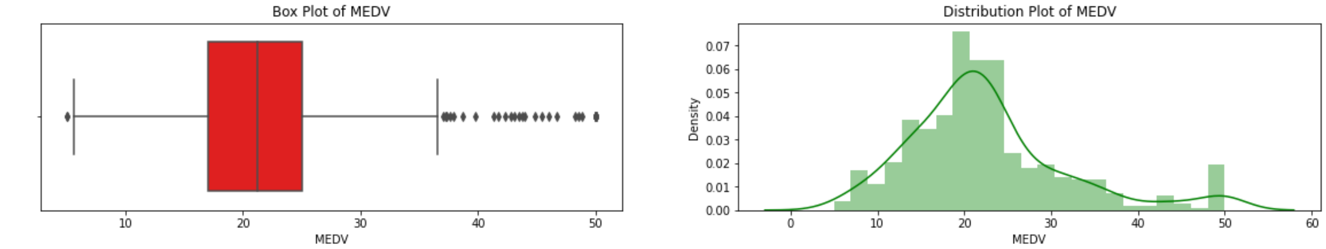
Double-click (or enter) to edit

3. Univariate and Multivariate Analysis

3.1 MEDV

```
#Box Plot and Distribution Plot for Dependent variable MEDV
plt.figure(figsize=(20,3))

plt.subplot(1,2,1)
sns.boxplot(df1.MEDV,color='red')
plt.title('Box Plot of MEDV')
```

```
plt.subplot(1,2,2)
sns.distplot(a=df1.MEDV,color='green')
plt.title('Distribution Plot of MEDV')
plt.show()
```



Write the inferences

Removing outliers

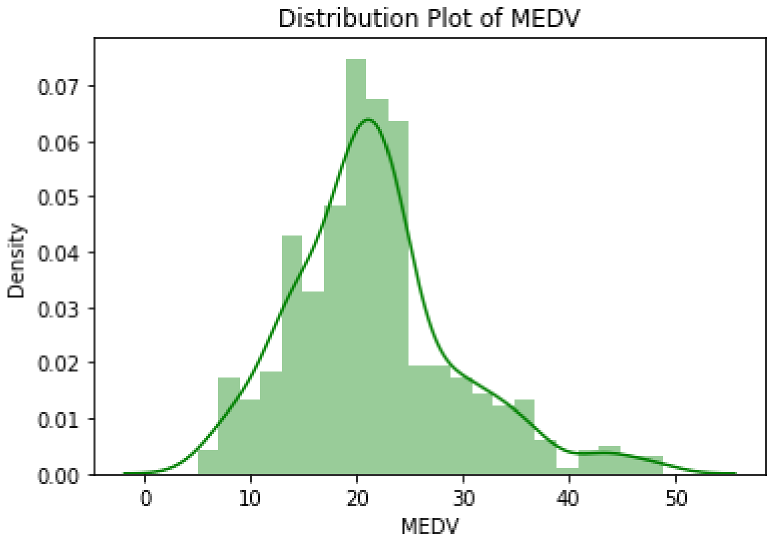
```
df2 = df1[~(df1['MEDV']==50)]
```

```
df2
```

	INDUS	NOX	RM	TAX	PTRATIO	LSTAT	MEDV
0	2.31	0.538	6.575	296.0	15.3	4.98	24.0
1	7.07	0.469	6.421	242.0	17.8	9.14	21.6
2	7.07	0.469	7.185	242.0	17.8	4.03	34.7
3	2.18	0.458	6.998	222.0	18.7	2.94	33.4
4	2.18	0.458	7.147	222.0	18.7	5.33	36.2
...	...	...	...	...	...	...	...
501	11.93	0.573	6.593	273.0	21.0	9.67	22.4
502	11.93	0.573	6.120	273.0	21.0	9.08	20.6
503	11.93	0.573	6.976	273.0	21.0	5.64	23.9
504	11.93	0.573	6.794	273.0	21.0	6.48	22.0
505	11.93	0.573	6.030	273.0	21.0	7.88	11.9

490 rows × 7 columns

```
sns.distplot(a=df2.MEDV,color='green')
plt.title('Distribution Plot of MEDV')
plt.show()
```




```
df2.max()
```

```
INDUS       27.740
NOX          0.871
RM           8.780
TAX        711.000
PTRATIO     22.000
LSTAT       37.970
MEDV        48.800
dtype: float64
```

```
print(f'Shape of dataset before removing Outliers: {df1.shape}')
#df2 = df1[~(df1['MEDV']==50)]
print(f'Shape of dataset after removing Outliers: {df2.shape}')
```

```
Shape of dataset before removing Outliers: (506, 7)
Shape of dataset after removing Outliers: (490, 7)
```

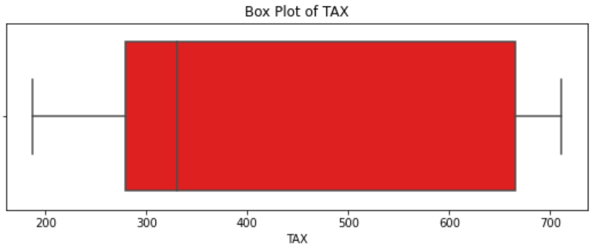
As we can see that we have deleted 16 rows from out dataset having MEDV = 50

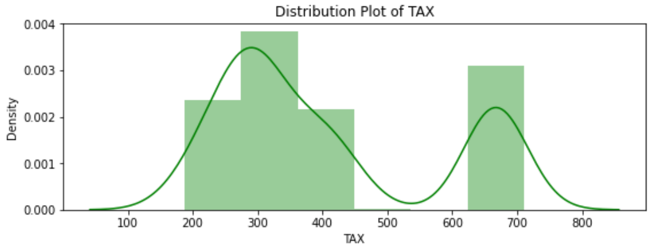
Repeat the same for rest of the features and write inferences about it.

3.2 TAX

```
#Box Plot and Distribution Plot for Dependent variable TAX
plt.figure(figsize=(20,3))

plt.subplot(1,2,1)
sns.boxplot(df1.TAX,color='red')
plt.title('Box Plot of TAX')
```

```
plt.subplot(1,2,2)
sns.distplot(a=df1.TAX,color='green')
plt.title('Distribution Plot of TAX')
plt.show()
```





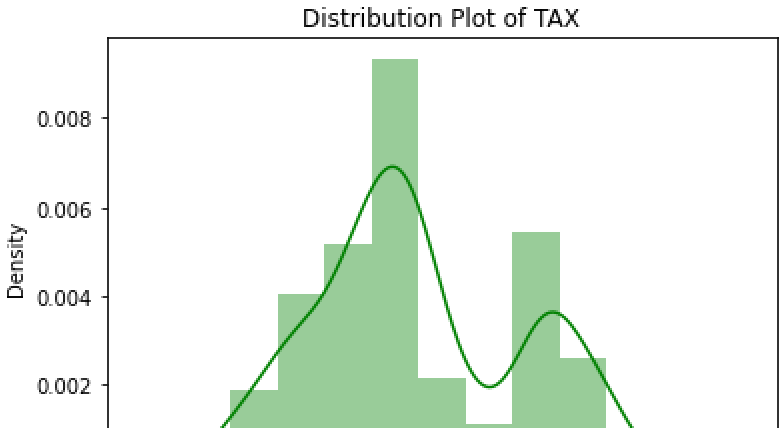

```
df2 = df1[~(df1['TAX']>=600)]
```

```
df2
```

	INDUS	NOX	RM	TAX	PTRATIO	LSTAT	MEDV
0	2.31	0.538	6.575	296.0	15.3	4.98	24.0
1	7.07	0.469	6.421	242.0	17.8	9.14	21.6
2	7.07	0.469	7.185	242.0	17.8	4.03	34.7
3	2.18	0.458	6.998	222.0	18.7	2.94	33.4
4	2.18	0.458	7.147	222.0	18.7	5.33	36.2
...	...	...	...	...	...	...	...
501	11.93	0.573	6.593	273.0	21.0	9.67	22.4
502	11.93	0.573	6.120	273.0	21.0	9.08	20.6
503	11.93	0.573	6.976	273.0	21.0	5.64	23.9
504	11.93	0.573	6.794	273.0	21.0	6.48	22.0
505	11.93	0.573	6.030	273.0	21.0	7.88	11.9

369 rows × 7 columns

```
sns.distplot(a=df2.TAX,color='green')
plt.title('Distribution Plot of TAX')
plt.show()
```




```
df2.max()
```

```
INDUS       25.650
NOX          0.871
RM           8.725
TAX        469.000
PTRATIO     22.000
LSTAT       34.410
MEDV        50.000
dtype: float64
```

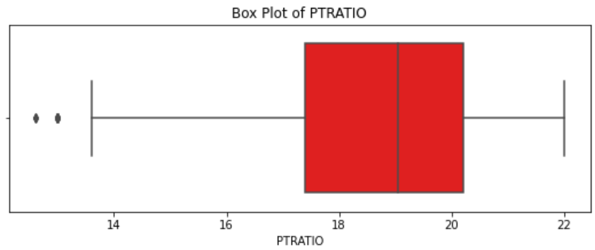
```
print(f'Shape of dataset before removing Outliers: {df1.shape}')
#df2 = df1[~(df1['TAX']>=600)]
print(f'Shape of dataset after removing Outliers: {df2.shape}')
```

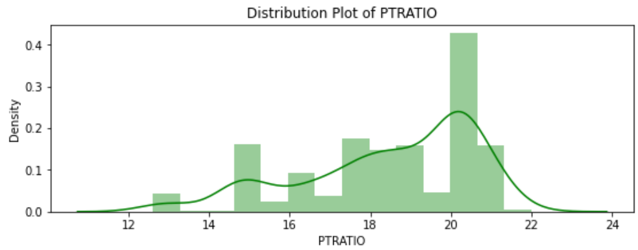
```
Shape of dataset before removing Outliers: (506, 7)
Shape of dataset after removing Outliers: (369, 7)
```

As we can see that we have deleted 137 rows from out dataset having TAX>=600

3.3 PTRATIO

```
#Box Plot and Distribution Plot for Dependent variable PTRATIO
plt.figure(figsize=(20,3))

plt.subplot(1,2,1)
sns.boxplot(df1.PTRATIO,color='red')
plt.title('Box Plot of PTRATIO')

plt.subplot(1,2,2)
sns.distplot(a=df1.PTRATIO,color='green')
plt.title('Distribution Plot of PTRATIO')
plt.show()
```





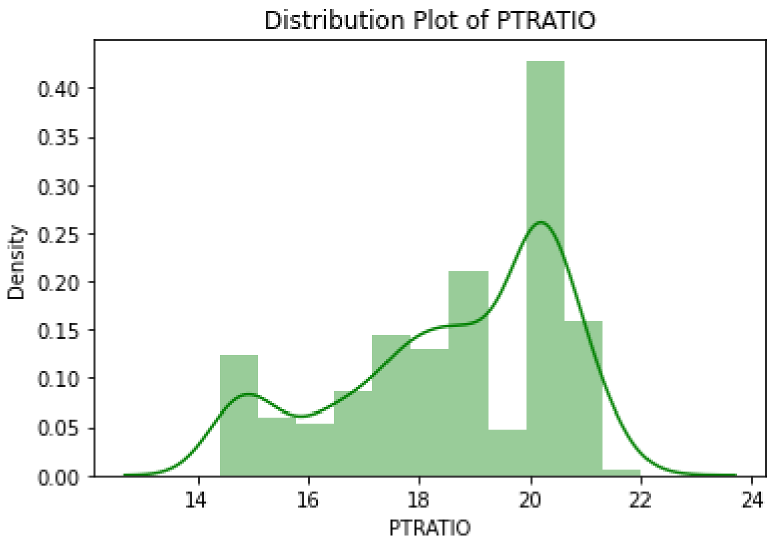

```
df2 = df1[~(df1['PTRATIO']<=14)]
```

```
df2
```

	INDUS	NOX	RM	TAX	PTRATIO	LSTAT	MEDV
0	2.31	0.538	6.575	296.0	15.3	4.98	24.0
1	7.07	0.469	6.421	242.0	17.8	9.14	21.6
2	7.07	0.469	7.185	242.0	17.8	4.03	34.7
3	2.18	0.458	6.998	222.0	18.7	2.94	33.4
4	2.18	0.458	7.147	222.0	18.7	5.33	36.2
...	...	...	...	...	...	...	...
501	11.93	0.573	6.593	273.0	21.0	9.67	22.4
502	11.93	0.573	6.120	273.0	21.0	9.08	20.6
503	11.93	0.573	6.976	273.0	21.0	5.64	23.9
504	11.93	0.573	6.794	273.0	21.0	6.48	22.0
505	11.93	0.573	6.030	273.0	21.0	7.88	11.9

490 rows × 7 columns

```
sns.distplot(a=df2.PTRATIO,color='green')
plt.title('Distribution Plot of PTRATIO')
plt.show()
```



```
df2.max()
```

```
INDUS       27.740
NOX          0.871
RM           8.780
TAX        711.000
PTRATIO     22.000
LSTAT       37.970
MEDV        50.000
dtype: float64
```

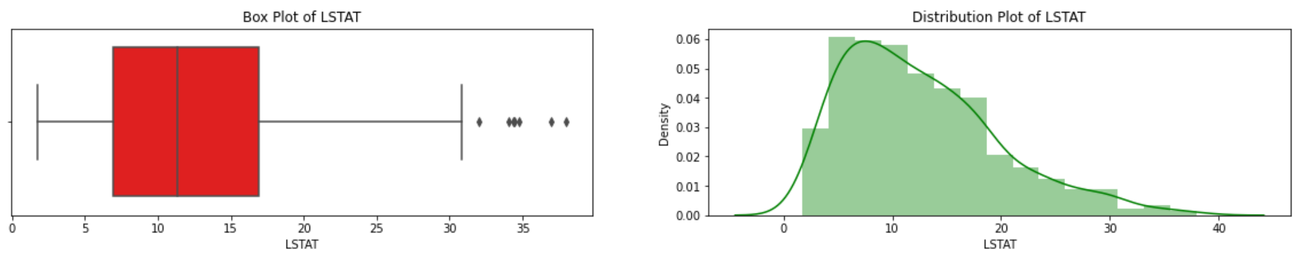
```
print(f'Shape of dataset before removing Outliers: {df1.shape}')
#df2 = df1[~(df1['PTRATIO']<=14)]
print(f'Shape of dataset after removing Outliers: {df2.shape}')
```

```
Shape of dataset before removing Outliers: (506, 7)
Shape of dataset after removing Outliers: (490, 7)
```

As we can see that we have deleted 16 rows from out dataset having PTRATIO<=14

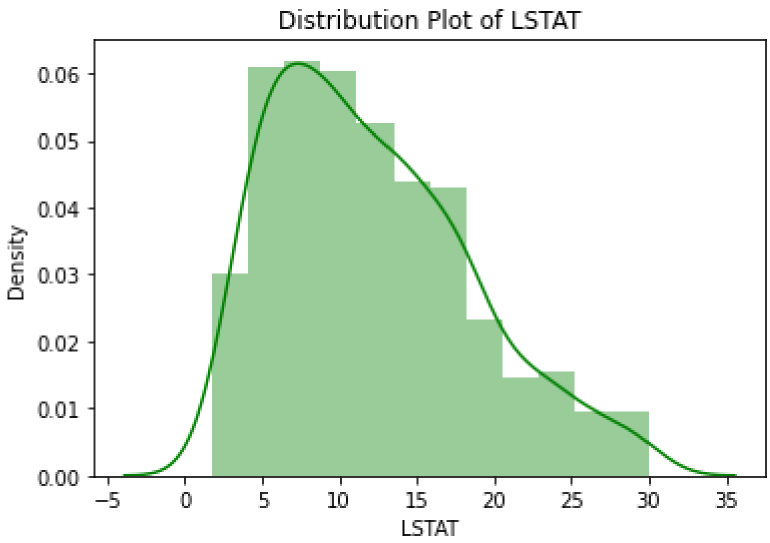
3.4 LSTAT

```
#Box Plot and Distribution Plot for Dependent variable LSTAT
plt.figure(figsize=(20,3))

plt.subplot(1,2,1)
sns.boxplot(df1.LSTAT,color='red')
plt.title('Box Plot of LSTAT')

plt.subplot(1,2,2)
sns.distplot(a=df1.LSTAT,color='green')
plt.title('Distribution Plot of LSTAT')
plt.show()
```




```
df2 = df1[~(df1['LSTAT']>=30)]
```

```
sns.distplot(a=df2.LSTAT,color='green')
plt.title('Distribution Plot of LSTAT')
plt.show()
```



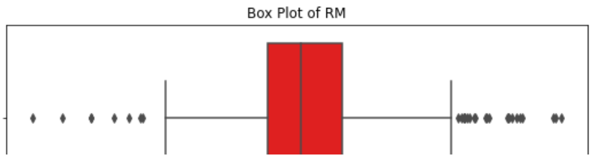

```
print(f'Shape of dataset before removing Outliers: {df1.shape}')
#df2 = df1[~(df1['LSTAT']>=30)]
print(f'Shape of dataset after removing Outliers: {df2.shape}')
```

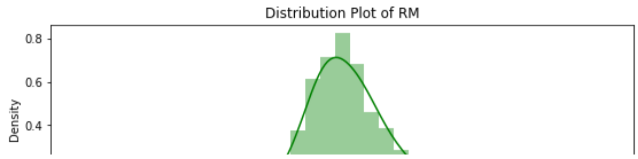
```
Shape of dataset before removing Outliers: (506, 7)
Shape of dataset after removing Outliers: (494, 7)
```

As we can see that we have deleted 12 rows from out dataset having LSTAT>=30

3.5 RM

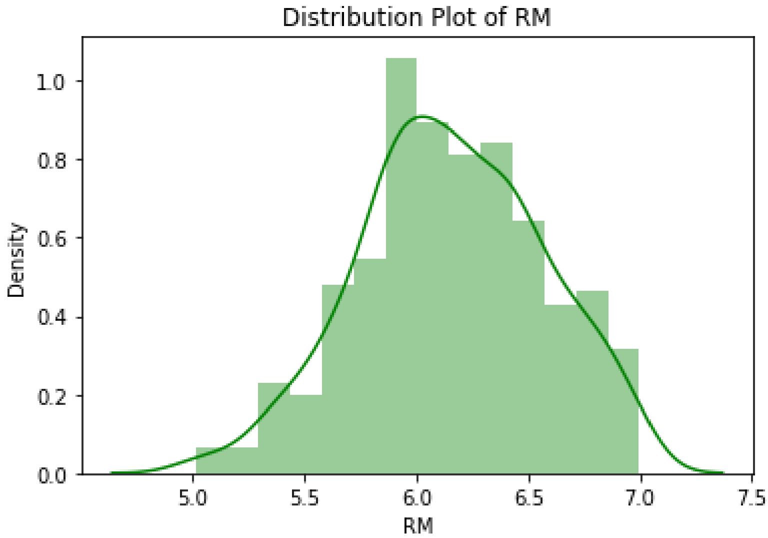
```
#Box Plot and Distribution Plot for Dependent variable RM
plt.figure(figsize=(20,3))

plt.subplot(1,2,1)
sns.boxplot(df1.RM,color='red')
plt.title('Box Plot of RM')

plt.subplot(1,2,2)
sns.distplot(a=df1.RM,color='green')
plt.title('Distribution Plot of RM')
plt.show()
```






```
df2 = df1[~(df1['RM']>=7)]
df3 = df2[~(df2['RM']<=5)]
```

```
sns.distplot(a=df3.RM,color='green')
plt.title('Distribution Plot of RM')
plt.show()
```




```
print(f'Shape of dataset before removing Outliers: {df1.shape}')
#df2 = df1[~(df1['RM']>=7)]
#df3 = df2[~(df2['RM']<=5)]
print(f'Shape of dataset after removing Outliers: {df3.shape}')
```

```
Shape of dataset before removing Outliers: (506, 7)
Shape of dataset after removing Outliers: (426, 7)
```

As we can see that we have deleted 80 rows from out dataset having RM>=7 & RM<=5