Experiment 1.1

Student Name-Pushpraj Roy **UID- 20BCS9866**
Branch- CSE **Section/Group- 617 A**
Semester- 5th **Subject Code- 20CSP-317**
Subject Name–Machine Learning Lab

AIM:- Exploratory data analysis (EDA).

OBJECTIVE:–To Understand the data i.e., Data is clean , it doesn't have any null values , missing values , remove noise , identify variables in dataset and relationship between variables to conclude the values.

Steps Involved:-

1.Treatment of Missing Values and Outliers (Preparation of the dataset for analysis by the treatment of the irregularities.

2.Draw meaningful patterns and insights with the help of data visualization to summarize their main characteristics.

S/W Requirement:- VS Code or Jupyter Notebook

INPUT AND OUTPUT:-

```
import pandas as pd import
numpy as np import
matplotlib.pyplot as plt import
seaborn as sns

df= pd.read_csv('housing.csv')
```

[4]
```python
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
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	24.0
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	21.6
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	34.7
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	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN	36.2

[5]
```python
df.tail()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	NaN	22.4
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	20.6
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	23.9
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	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	7.88	11.9

[7]
```python
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATI
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.000000	506.000000	506.000000	506.00000
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.795043	9.549407	408.237154	18.45553
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.105710	8.707259	168.537116	2.16494
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.60000
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.100175	4.000000	279.000000	17.40000
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.207450	5.000000	330.000000	19.05000
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.188425	24.000000	666.000000	20.20000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.00000

Getting its dimension df.shape

```
# Getting its dimension
df.shape
```
[8] Python

```
(506, 14)
```

obtain the missing values present in the given raw Housing Data df.isnull().sum()

```
# obtain the missing values present in the given raw Housing Data
df.isnull().sum()
```
[9] Python

```
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
```

getting the column names of the dataset df.columns

```
# getting the column names of the dataset
df.columns
```
[10] Python

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

Importing the visualization package of Python
Detection of outliers among all variables

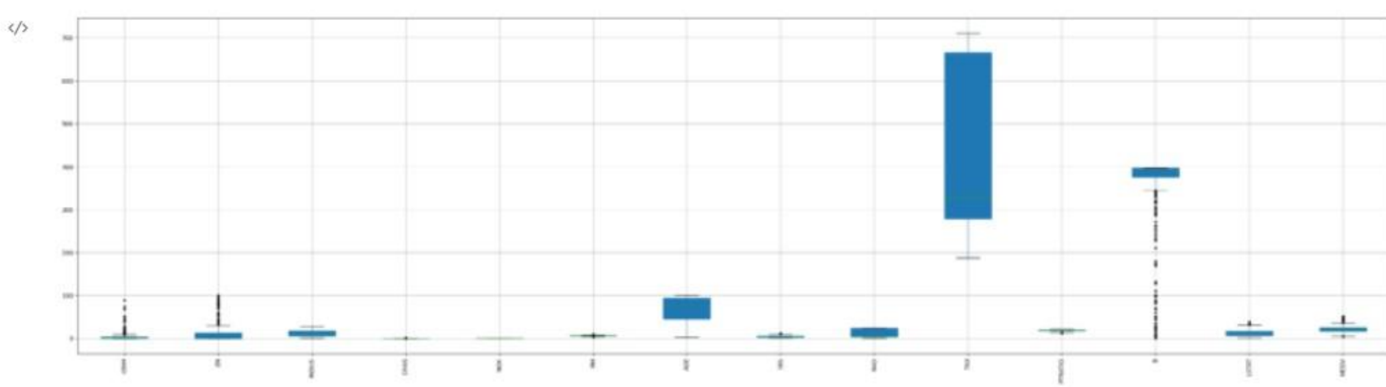
import matplotlib.pyplot as plt import
seaborn as sns %matplotlib inline

plt.subplots(figsize=(39,10))
df.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)

```
# Importing the visualization package of Python
# Detection of outliers among all variables
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.subplots(figsize=(39,10))
df.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

[11] Python

```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 [Text(1, 0, 'CRIM'),
  Text(2, 0, 'ZN'),
  Text(3, 0, 'INDUS'),
  Text(4, 0, 'CHAS'),
  Text(5, 0, 'NOX'),
  Text(6, 0, 'RM'),
  Text(7, 0, 'AGE'),
  Text(8, 0, 'DIS'),
  Text(9, 0, 'RAD'),
  Text(10, 0, 'TAX'),
  Text(11, 0, 'PTRATIO'),
  Text(12, 0, 'B'),
  Text(13, 0, 'LSTAT'),
  Text(14, 0, 'MEDV')])
```



#PHASE 1: TREATMENT OF MISSING VALUES
For first category: "cat_mv_out"

cat_mv = pd.concat([df["CHAS"]],axis=1) cat_mv

```python
#PHASE 1: TREATMENT OF MISSING VALUES
# For first category: "cat_mv_out"
cat_mv = pd.concat([df["CHAS"]],axis=1)
cat_mv
```

[14] Python

	CHAS
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
501	0.0
502	0.0
503	0.0
504	0.0
505	0.0

506 rows × 1 columns

cat_mv.isnull().sum()

```python
cat_mv.isnull().sum()
```

[15] Python

```
CHAS    20
dtype: int64
```

cat_mv.mode()

```python
cat_mv.mode()
```

[16] Python

	CHAS
0	0.0

Replacing the missing values with mode(value 0) to this categorical variable

replace nan value to zero(mode = 0) cat_mv.replace(np.nan, 0, inplace=True)

After replacing with mode(Value = 0), now there is no missing values in this categorical variable cat_mv.isnull().sum()

```
# After replacing with mode(Value = 0), now there is no missing values in this categorical variable
cat_mv.isnull().sum()
[18]                                                                    Python
...  CHAS    0
     dtype: int64
```

dimension (506 Observations and 1 column) cat_mv.shape

```
# dimension (506 Observations and 1 column)
cat_mv.shape
[19]                                                                    Python
...  (506, 1)
```

# For the second category: "num_mv_out" means Numerical variables containing missing values and outliers too
num_mv_out = pd.concat([df["CRIM"], df["ZN"], df["LSTAT"]],axis=1) num_mv_out.isnull().sum()

```
num_mv_out.isnull().sum()
[21]                                                                    Python
...  CRIM     20
     ZN       20
     LSTAT    20
     dtype: int64
```

# Describe the numericaldata which as missing values and outliers.
num_mv_out.describe()

```
# Describe the  numericaldata which as missing values and outliers.
num_mv_out.describe()
```

[22]

	CRIM	ZN	LSTAT
count	486.000000	486.000000	486.000000
mean	3.611874	11.211934	12.715432
std	8.720192	23.388876	7.155871
min	0.006320	0.000000	1.730000
25%	0.081900	0.000000	7.125000
50%	0.253715	0.000000	11.430000
75%	3.560263	12.500000	16.955000
max	88.976200	100.000000	37.970000

# Replacing the missing values with median of its variables ("num_mv_out")
num_mv_out = num_mv_out.fillna(num_mv_out.median())
# Now, "num_mv_out" has no missing values num_mv_out.isnull().sum()

```
# Now, "num_mv_out" has no missing values
num_mv_out.isnull().sum()
```

[24]

```
CRIM     0
ZN       0
LSTAT    0
dtype: int64
```

num_mv_out.shape

```
num_mv_out.shape
```

[25]

```
(506, 3)
```

# For the third category: "num_mv_noOut" means Numerical variables containing missing values but "no outliers"
num_mv_noOut = pd.concat([df["INDUS"], df["AGE"]],axis=1) num_mv_noOut

	INDUS	AGE
0	2.31	65.2
1	7.07	78.9
2	7.07	61.1
3	2.18	45.8
4	2.18	54.2
...	...	...
501	11.93	69.1
502	11.93	76.7
503	11.93	91.0
504	11.93	89.3
505	11.93	NaN

506 rows × 2 columns

#checking the variables ; is there any null value present or not num_mv_noOut.isnull().sum()

```
INDUS    20
AGE      20
dtype: int64
```

Replacing the missing values with mean of its variable ("num_mv_noOut")
this category doesn't have outliers but having missing values in the two variables
num_mv_noOut = num_mv_noOut.fillna(num_mv_noOut.mean())

Now, this cateory ("num_mv_noOut") has no missing values num_mv_noOut.isnull().sum()

```
INDUS    0
AGE      0
dtype: int64
```

#PHASE 2: TREATMENT OF OUTLIERS
For assigning or concatenating all the variables including with six treated missing values variables into a dataset

df1 = pd.concat([cat_mv,num_mv_out, num_mv_noOut, df["RM"], df["DIS"], df["PTRATIO"], df["B"], df["MEDV"], df["NOX"], df["RAD"], df["TAX"]],axis=1)

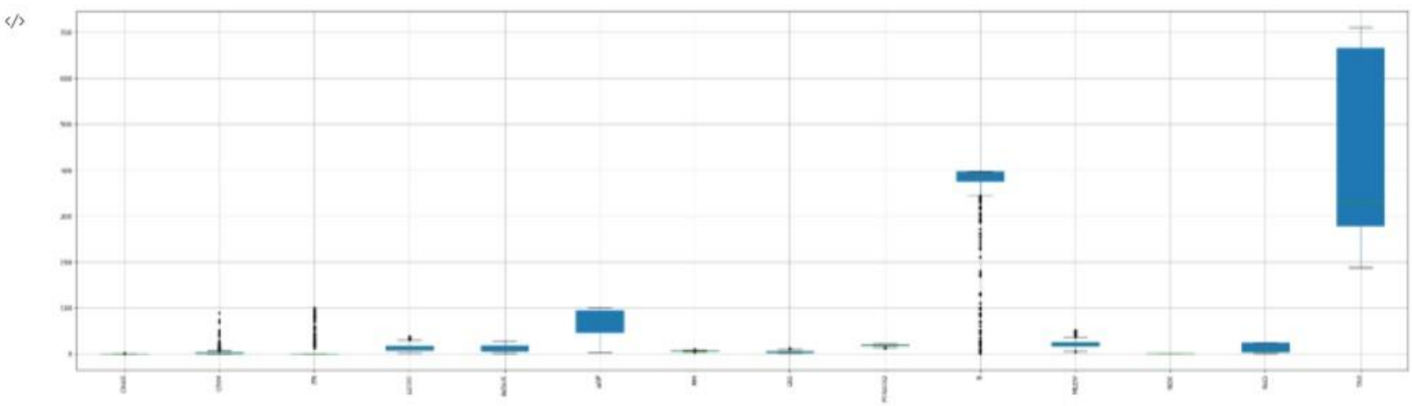
No missing values after merging all variables df1.isnull().sum()

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

```python
# Boxplot for all variables
plt.subplots(figsize=(37,10))
df1.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

[35] Python

```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 [Text(1, 0, 'CHAS'),
  Text(2, 0, 'CRIM'),
  Text(3, 0, 'ZN'),
  Text(4, 0, 'LSTAT'),
  Text(5, 0, 'INDUS'),
  Text(6, 0, 'AGE'),
  Text(7, 0, 'RM'),
  Text(8, 0, 'DIS'),
  Text(9, 0, 'PTRATIO'),
  Text(10, 0, 'B'),
  Text(11, 0, 'MEDV'),
  Text(12, 0, 'NOX'),
  Text(13, 0, 'RAD'),
  Text(14, 0, 'TAX')])
```



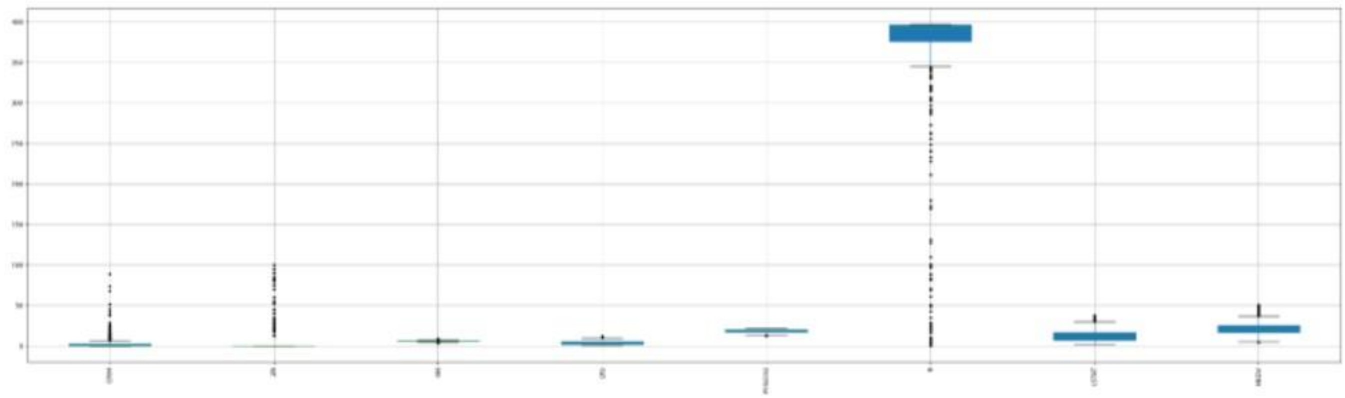
#Now, It's time for treatment of outliers
#1.num_out = Numerical variables containing outliers (Missing values will be treated with median)---
"CRIM", "ZN", "RM", "DIS", "PTRATIO", "B", "LSTAT", "MEDV"

num_out = pd.concat([df1["CRIM"], df1["ZN"], df1["RM"], df1["DIS"], df1["PTRATIO"], df1["B"], df1["LSTAT"], df1["MEDV"]],axis=1)

Detecting outliers in "cat_out"

plt.subplots(figsize=(37,10))
num_out.boxplot(patch_artist=True, sym="k.") plt.xticks(rotation=90)

```
(array([1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(1, 0, 'CRIM'),
  Text(2, 0, 'ZN'),
  Text(3, 0, 'RM'),
  Text(4, 0, 'DIS'),
  Text(5, 0, 'PTRATIO'),
  Text(6, 0, 'B'),
  Text(7, 0, 'LSTAT'),
  Text(8, 0, 'MEDV')])
```



```
# Getting the basic statistical summary of those variables containing outliers
num_out.describe()
```

[39] Python

	CRIM	ZN	RM	DIS	PTRATIO	B	LSTAT	MEDV
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.479140	10.768775	6.284634	3.795043	18.455534	356.674032	12.664625	22.532806
std	8.570832	23.025124	0.702617	2.105710	2.164946	91.294864	7.017219	9.197104
min	0.006320	0.000000	3.561000	1.129600	12.600000	0.320000	1.730000	5.000000
25%	0.083235	0.000000	5.885500	2.100175	17.400000	375.377500	7.230000	17.025000
50%	0.253715	0.000000	6.208500	3.207450	19.050000	391.440000	11.430000	21.200000
75%	2.808720	0.000000	6.623500	5.188425	20.200000	396.225000	16.570000	25.000000
max	88.976200	100.000000	8.780000	12.126500	22.000000	396.900000	37.970000	50.000000

Detecting and Removing Outliers
Inter Quartile Range (IQR) is the difference between the 3rd Quartile and the first Quartile # The data points which fall below Q1 – 1.5 IQR or above Q3 + 1.5 IQR are outliers.

def detect_outlier(feature): Q1
= np.percentile(feature, 25)
Q3 = np.percentile(feature, 75)

```
    IQR = Q3 - Q1     IQR
*= 1.5     minimum = Q1
- IQR      maximum =
Q3 + IQR  flag = False

 if(minimum > np.min(feature)):
flag = True     if(maximum <
np.max(feature)):        flag = True

    return flag
```

```
#Using tukey method to remove outliers. Whiskers are set at 1.5 times Interquartile Range (IQR).
# Any value beyond the acceptance range are considered as outliers.
#Replacing the outliers with the median value of that feature
#Why replacing with median value?
#As the mean value is highly influenced by the outliers, it is advised to replace the outliers with the median value.
```

Python

```
def  remove_outlier(feature):
    Q1 = np.percentile(num_out[feature], 25)
    Q3 = np.percentile(num_out[feature], 75)
    IQR = Q3 - Q1
    IQR *= 1.5

    minimum = Q1 - IQR # the acceptable minimum value
maximum = Q3 + IQR # the acceptable maximum value

    median = num_out[feature].median()

    num_out.loc[num_out[feature] < minimum, feature] = median
num_out.loc[num_out[feature] > maximum, feature] = median

# taking all the column

num_out = num_out.iloc[:, : ] for i in
range(len(num_out.columns)):
        remove_outlier(num_out.columns[i])
```

In "num_out" matrix, it contains all varibles num_out
= num_out.iloc[:, :]

```python
num_out
```

[44] Python

	CRIM	ZN	RM	DIS	PTRATIO	B	LSTAT	MEDV
0	0.00632	0.0	6.575	4.0900	15.3	396.90	4.98	24.0
1	0.02731	0.0	6.421	4.9671	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.185	4.9671	17.8	392.83	4.03	34.7
3	0.03237	0.0	6.998	6.0622	18.7	394.63	2.94	33.4
4	0.06905	0.0	7.147	6.0622	18.7	396.90	11.43	36.2
...	...	...	...	...	...	...	...	...
501	0.06263	0.0	6.593	2.4786	21.0	391.99	11.43	22.4
502	0.04527	0.0	6.120	2.2875	21.0	396.90	9.08	20.6
503	0.06076	0.0	6.976	2.1675	21.0	396.90	5.64	23.9
504	0.10959	0.0	6.794	2.3889	21.0	393.45	6.48	22.0
505	0.04741	0.0	6.030	2.5050	21.0	396.90	7.88	11.9

506 rows × 8 columns

```python
# This shows that these are the variables from "num_out" which contain Outliers
for i in range(len(num_out.columns)):
    if(detect_outlier(num_out[num_out.columns[i]])):
        print(num_out.columns[i], "Contains Outlier")
```

[45] Python

```
CRIM Contains Outlier
RM Contains Outlier
B Contains Outlier
LSTAT Contains Outlier
MEDV Contains Outlier
```

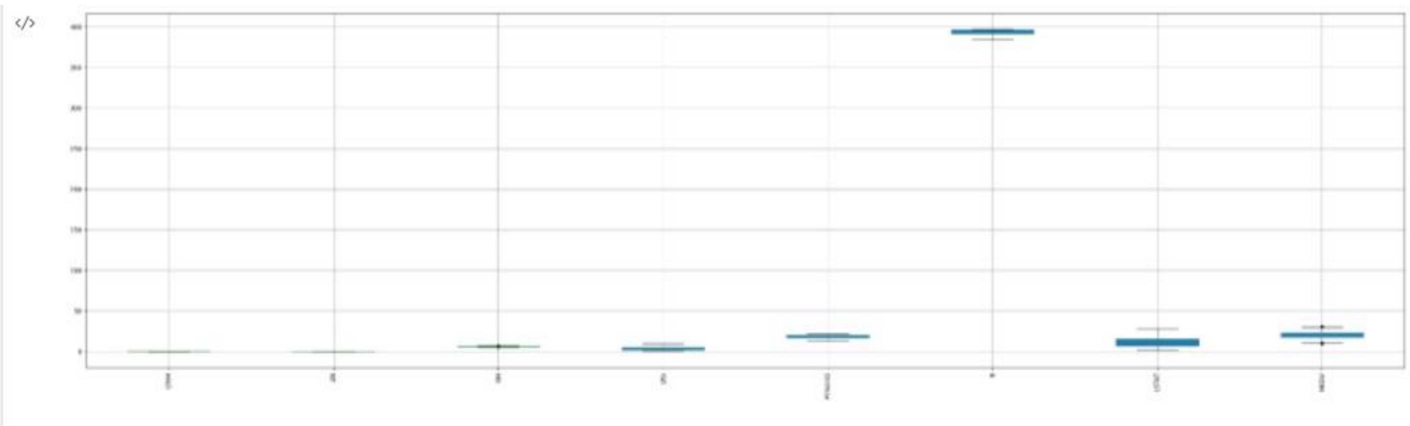
```python
# Removing the outliers
for i in range (3):
    for i in range(len(num_out.columns)):
        remove_outlier(num_out.columns[i])
```

[46] Python

```python
# After removing outliers, the following boxplots of each variable from "num_out" show, they have no more outliers
plt.subplots(figsize=(37,10))
num_out.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

[47] Python

```
(array([1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(1, 0, 'CRIM'),
  Text(2, 0, 'ZN'),
  Text(3, 0, 'RM'),
  Text(4, 0, 'DIS'),
  Text(5, 0, 'PTRATIO'),
  Text(6, 0, 'B'),
  Text(7, 0, 'LSTAT'),
  Text(8, 0, 'MEDV')])
```



```python
# Finally, concatenating all variables after treatment of outliers with those varibales
# that have no outliers into a dataset
final_df = pd.concat([num_out, df1["CHAS"], df1["INDUS"], df1["NOX"], df1["AGE"], df1["RAD"], df1["TAX"]],axis=1)
```

[48] Python

```python
#After treatment of missing values as well as outliers
#The dataset is now ready for further analysis
```

[ ] Python

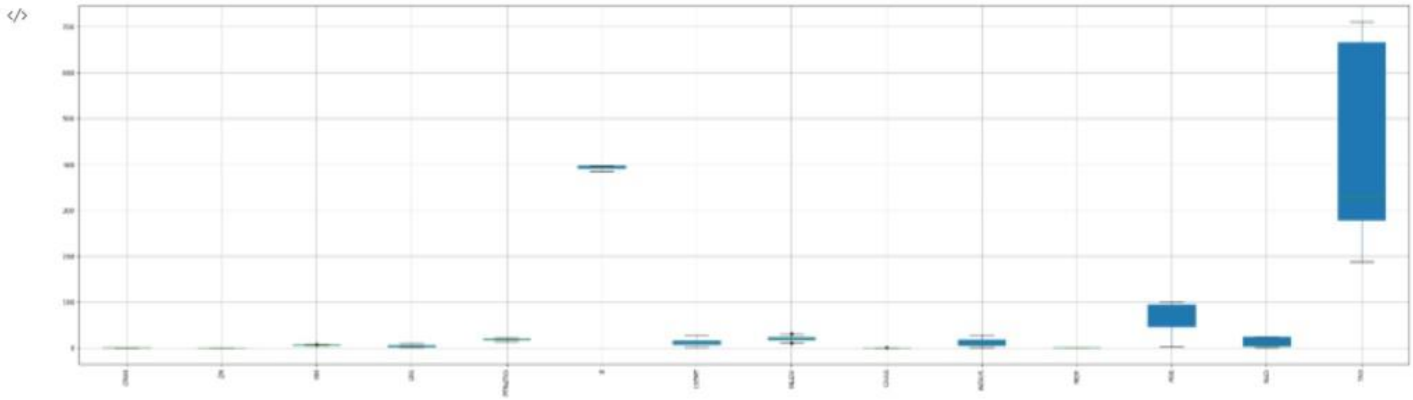
```python
final_df
```

[49] Python

	CRIM	ZN	RM	DIS	PTRATIO	B	LSTAT	MEDV	CHAS	INDUS	NOX	AGE	RAD	TAX
0	0.00632	0.0	6.575	4.0900	15.3	396.90	4.98	24.0	0.0	2.31	0.538	65.200000	1	296
1	0.02731	0.0	6.421	4.9671	17.8	396.90	9.14	21.6	0.0	7.07	0.469	78.900000	2	242
2	0.02729	0.0	7.185	4.9671	17.8	392.83	4.03	21.2	0.0	7.07	0.469	61.100000	2	242
3	0.03237	0.0	6.998	6.0622	18.7	394.63	2.94	21.2	0.0	2.18	0.458	45.800000	3	222
4	0.06905	0.0	7.147	6.0622	18.7	396.90	11.43	21.2	0.0	2.18	0.458	54.200000	3	222
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	6.593	2.4786	21.0	391.99	11.43	22.4	0.0	11.93	0.573	69.100000	1	273
502	0.04527	0.0	6.120	2.2875	21.0	396.90	9.08	20.6	0.0	11.93	0.573	76.700000	1	273
503	0.06076	0.0	6.976	2.1675	21.0	396.90	5.64	23.9	0.0	11.93	0.573	91.000000	1	273
504	0.10959	0.0	6.794	2.3889	21.0	393.45	6.48	22.0	0.0	11.93	0.573	89.300000	1	273
505	0.04741	0.0	6.030	2.5050	21.0	396.90	7.88	11.9	0.0	11.93	0.573	68.518519	1	273

506 rows × 14 columns

Boxplot for the final dataset plt.subplots(figsize=(37,10)) final_df.boxplot(patch_artist=True, sym="k.") plt.xticks(rotation=90)

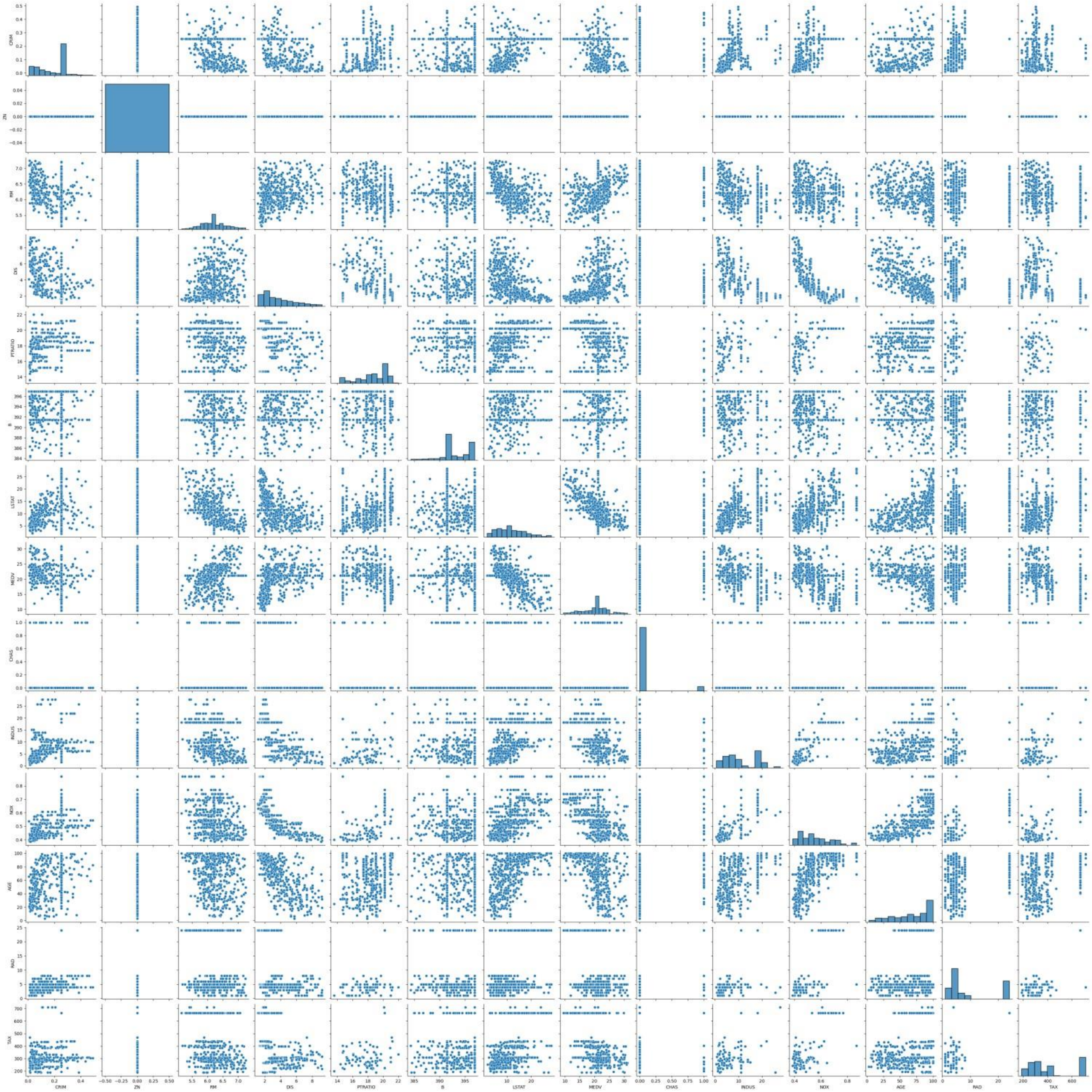


the heatmap also shows the same things and interpretations which earlier correlation matrix has been shown
fig, ax = plt.subplots(figsize=(37,13))
correlation_matrix = final_df.corr().round(2) #
annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)

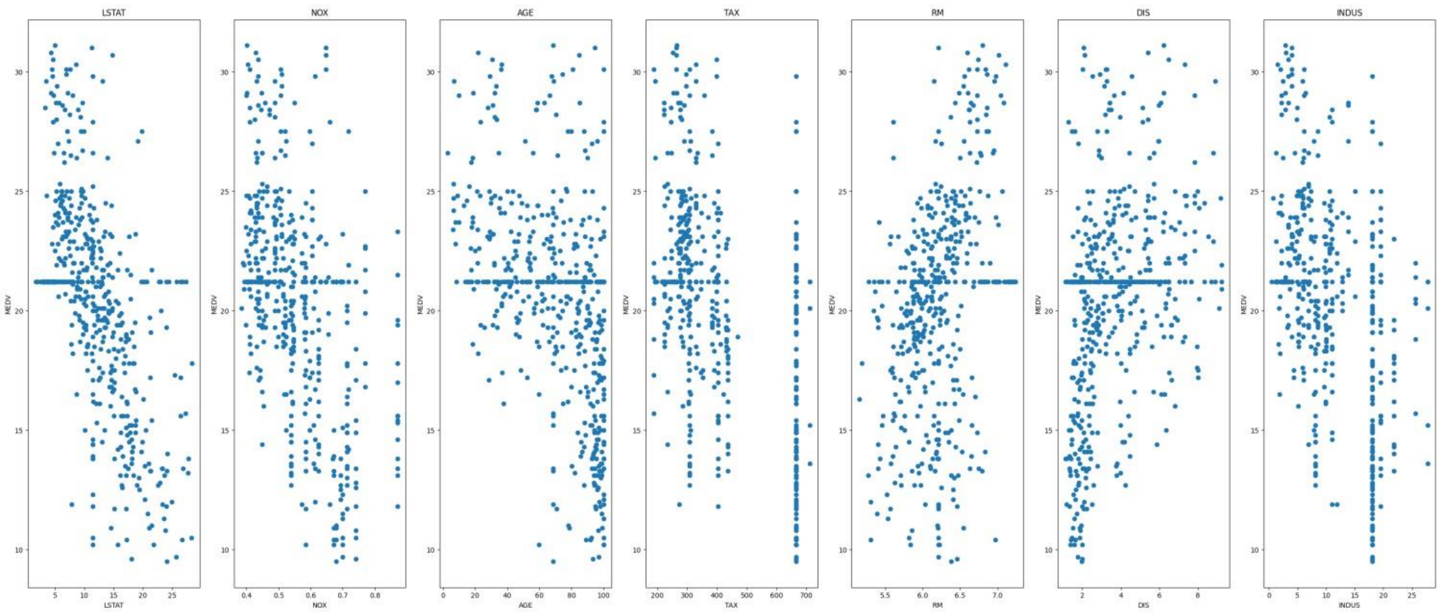
<AxesSubplot:>

Here, Correlation matrix shows:
the relationship among explanatory variables as well as,
the relationship between the dependent varibale with each of the explanatory variables

sns.pairplot(final_df)

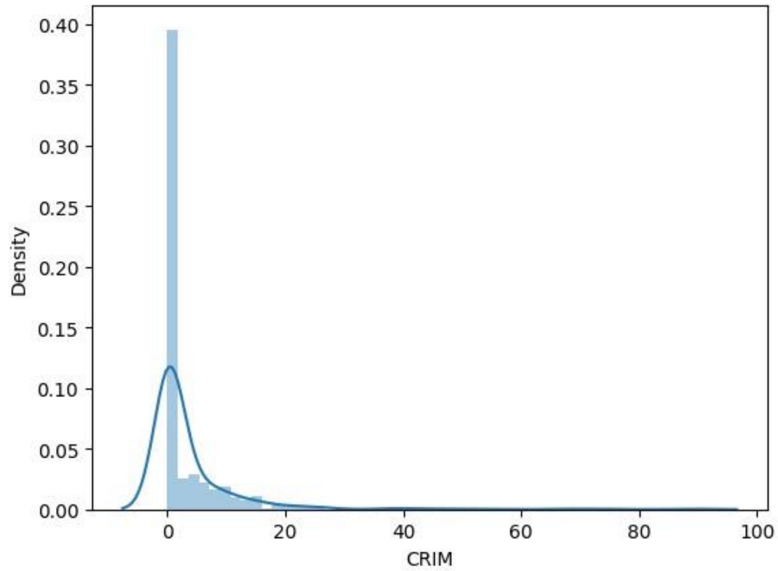

CRIM
ZN
RM
DIS
PTRATIO
B
LSTAT
MEDV
CHAS
INDUS
NOX
AGE
RAD
TAX

```
#scatter plot to see how these features RAD, RM ,DIS, LSTAT, NOX, AGE, TAX, INDUS vary with
Target variable (MEDV)
plt.figure(figsize=(37,15))

features = ['LSTAT','NOX','AGE','TAX','RM','DIS','INDUS'] target
= final_df['MEDV']

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




```
sns.distplot(df['CRIM'])
```

<AxesSubplot:xlabel='CRIM', ylabel='Density'>



Learning outcomes (What I have learnt) -

1. Identify the faulty points so that we can clean the data.
2. How to deal with missing values of variables (Columns) in dataset.
3. To Deal with Outliers.
4. To find Relationship between different variables and map different type of Graphs.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			