WORKSHEET – 1

Name: Yash Kumar
Branch: CSE
Semester: 5th
Subject: ML Lab

UID: 20BCS9256
Section/Group: 616 'B'
Date of Performance: 18/08/2022

Aim: Implement Exploratory Data Analysis on any data set.

Software used : Google collab

#Importing and uploading data set.

```
[1] # importing the basic packages of Python and dataset
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
#uploading data set
df = pd.read_csv("HousingData.csv")
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

#Displaying details

```
# getting the details of raw Housing Data
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
```

#Getting first 10 rows

```
[6] # Getting the first 10 rows of the dataset
df.head(10)
```

	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	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	NaN	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	NaN	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9

#Getting basic summary

```
[7] # Get the basic statistical summary of the variables present in the given raw data
    df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.000000	506.000000	506.000000	506.000000	506.0000
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.795043	9.549407	408.237154	18.455534	356.6740
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.105710	8.707259	168.537116	2.164946	91.2948
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.3200
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.100175	4.000000	279.000000	17.400000	375.3775
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.207450	5.000000	330.000000	19.050000	391.4400
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.188425	24.000000	666.000000	20.200000	396.2250
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.9000

#Getting dimension and obtain missing values

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

(506, 14)
```

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

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 columns

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

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

#Importing visualization package

```
[11] # Importing the visualization package of Python
     import matplotlib.pyplot as plt
     import seaborn as sns
     # Detection of outliers among all variables
```

PHASE 1: TREATMENT OF MISSING VALUES

Separate all six variables contain missing values into three groups on the basis of the presence of outliers.

cat_mv = Categorical variable containing missing values (Missing values will be treated with mode) --- "CHAS"

num_mv_out = Numerical variables containing missing values and outliers too (Missing values will be treated with median) --- "CRIM", "ZN”,"LSTAT"

num_mv_noOut = Numerical variables containing missing values but "no outliers" (Missing values will be treated with mean) --- "INDUS", "AGE"

#Initiating treatment of missing value

Initiate the treatment of Missing Values

```
[13] # For first category: "cat_mv_out"
     cat_mv = pd.concat([df["CHAS"]],axis=1)
     cat_mv
```

	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

```
[14] cat_mv.isnull().sum()

CHAS    20
dtype: int64
```

```
[15] cat_mv.mode()
```

	CHAS
0	0.0

```
[16] # 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)
```

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

CHAS    0
dtype: int64
```

```
[18] # dimension (506 Observations and 1 column)
     cat_mv.shape

     (506, 1)

[20] # 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()

     CRIM     20
     ZN       20
     LSTAT    20
     dtype: int64

[21] # Replacing the missing values with median of its variables ("num_mv_out")
     num_mv_out = num_mv_out.fillna(num_mv_out.median())

[22] # Now, "num_mv_out" has no missing values
     num_mv_out.isnull().sum()

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

```
[23] num_mv_out.shape

     (506, 3)

[25] # 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

#Final stage has no missing values

```
[26] num_mv_noOut.isnull().sum()

INDUS    20
AGE      20
dtype: int64

[27] # 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

After treatment of missing values, the dataset will have only outliers problems. So, the next treatment will be for outliers. Now, assign a dataset that will contain all 14 variables including the above three category ("Treated Missing Values" Variables). Finally, split this dataset into three categories. But the thing is, Only the first category will be focused here because the first category contains outliers. The second and third categories have no outliers.

num_out = Numerical variables containing outliers (Missing values will be treated with median) --- "CRIM", "ZN", "RM", "DIS", "PTRATIO", "B", "LSTAT", "MEDV"

num_noOut = Numerical variables containing "no outliers" (Missing values will be treated with mean) --- "INDUS", "NOX", "AGE", "RAD", "TAX"

cat_out = Categorical variable containing no outliers --- "CHAS" ----- n this variable, the observation is either 1 or 0

PHASE 2: TREATMENT OF OUTLIERS

```
[28] # 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"]],
```

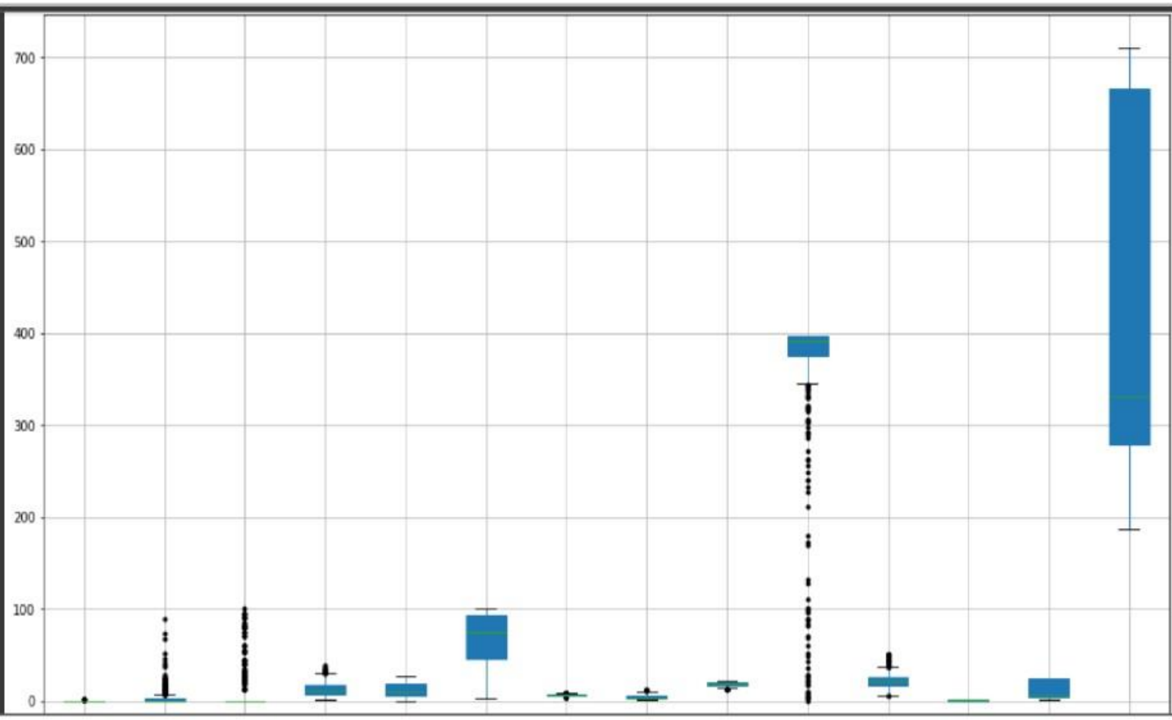
```
[29] # No missing values after merging all variables
     df1.isnull().sum()

     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
```

#Boxplot

```
[30] # Boxplot for all variables
     plt.subplots(figsize=(17,10))
     df1.boxplot(patch_artist=True, sym="k.")
     plt.xticks(rotation=90)
```

[30]



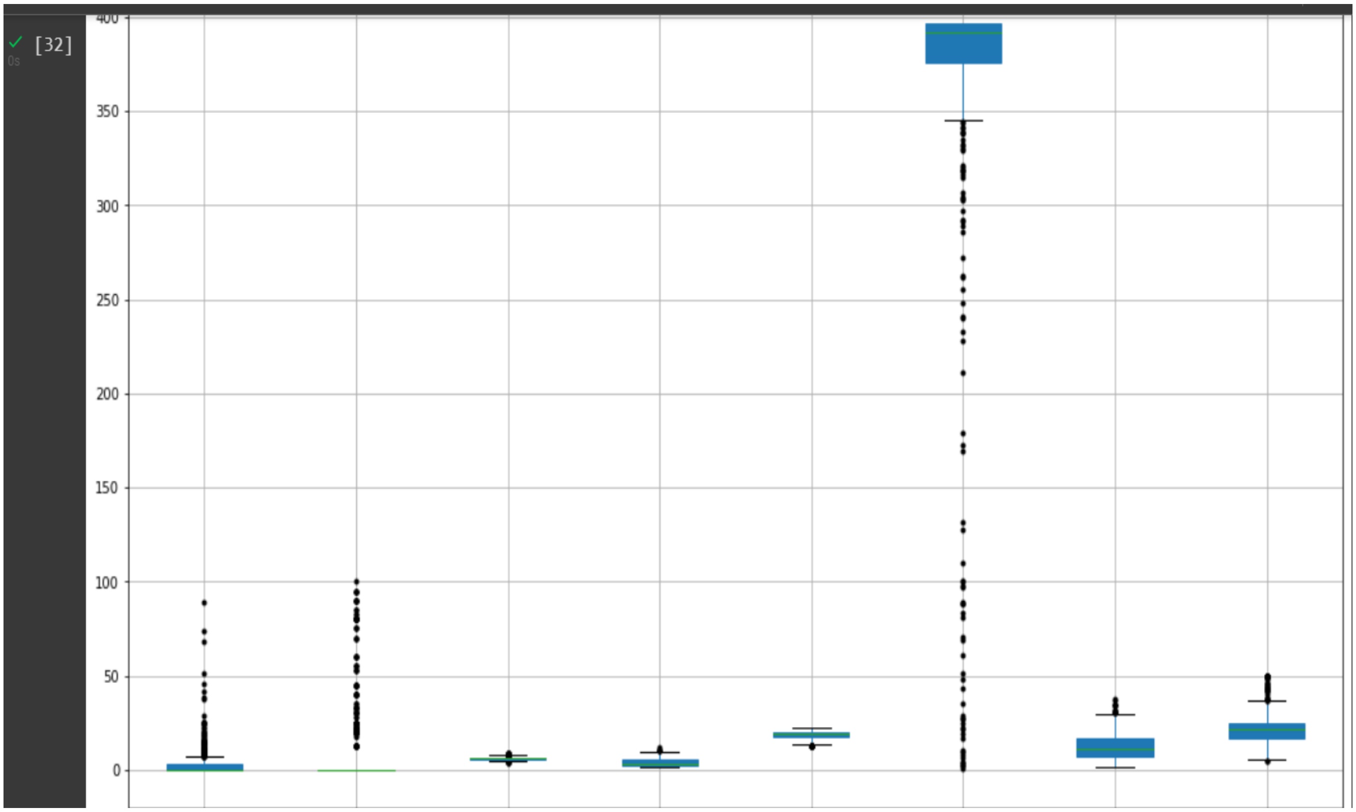

```
[31] num_out = pd.concat([df1["CRIM"], df1["ZN"], df1["RM"], df1["DIS"], df1["PTRATIO"], df1["B"], df1["LSTAT"], df1["MEDV"]],axis=1)
     num_out
```

	CRIM	ZN	RM	DIS	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	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

#Boxplot after detecting outliers

```
[32] # Detecting outliers in "cat_out"
     plt.subplots(figsize=(17,10))
     num_out.boxplot(patch_artist=True, sym="k.")
     plt.xticks(rotation=90)
```



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

	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

```
[34] # 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
```

```
[35] 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
```

```
[36] # taking all the column

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

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

	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

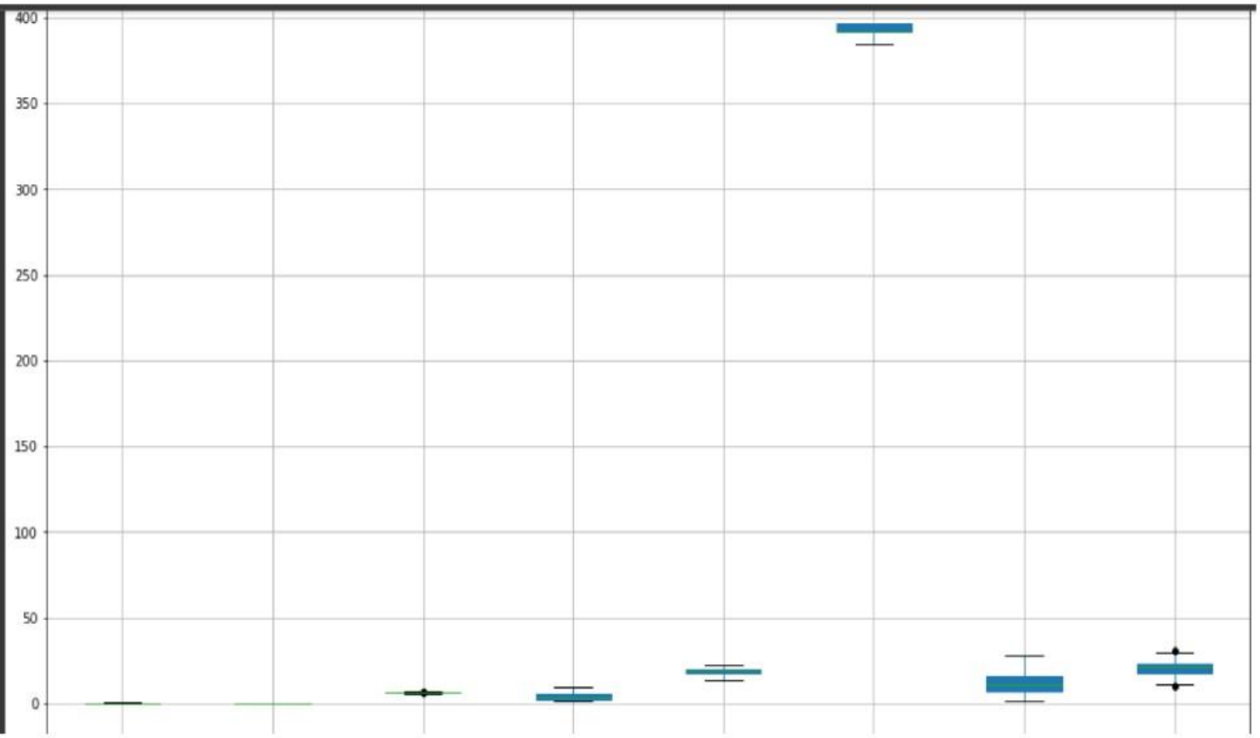
[40]
```
# This shows that these are the variables from "num_out" which contain Outliers
for i in range(len(num_out.columns)):
    if(detect_outlier(num_out[num_out.columns[i]])):
        print(num_out.columns[i], "Contains Outlier")
```

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

[41]
```
# Removing the outliers
for i in range (3):
    for i in range(len(num_out.columns)):
        remove_outlier(num_out.columns[i])
```

[42]
```
# After removing outliers, the following boxplots of each variable from "num_out" show, they have no more outliers
plt.subplots(figsize=(17,10))
num_out.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

[42]



```
[43] # 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)
```

Analysis

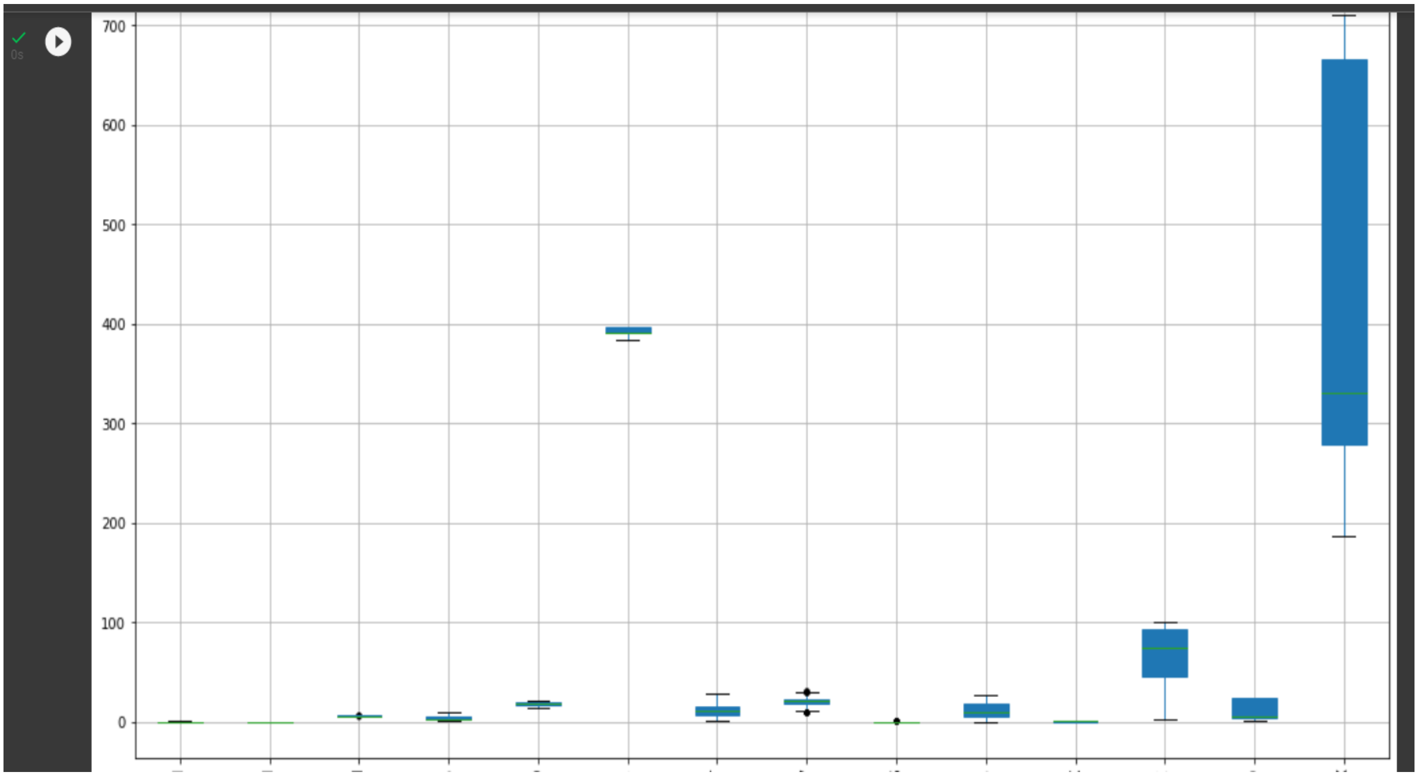
```
[44] final_df
```

	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

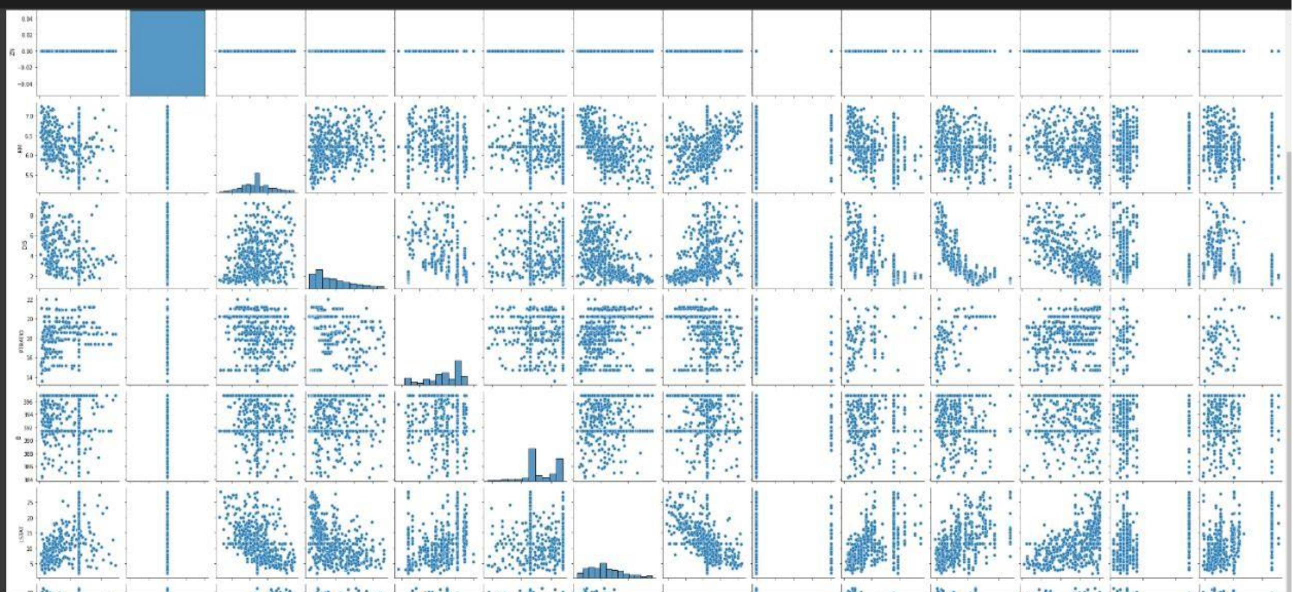
```python
# Boxplot for the final dataset
plt.subplots(figsize=(17,10))
final_df.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

#Final boxplot



#Pairplot

```
[46] #pairplot
     sns.pairplot(final_df)
```

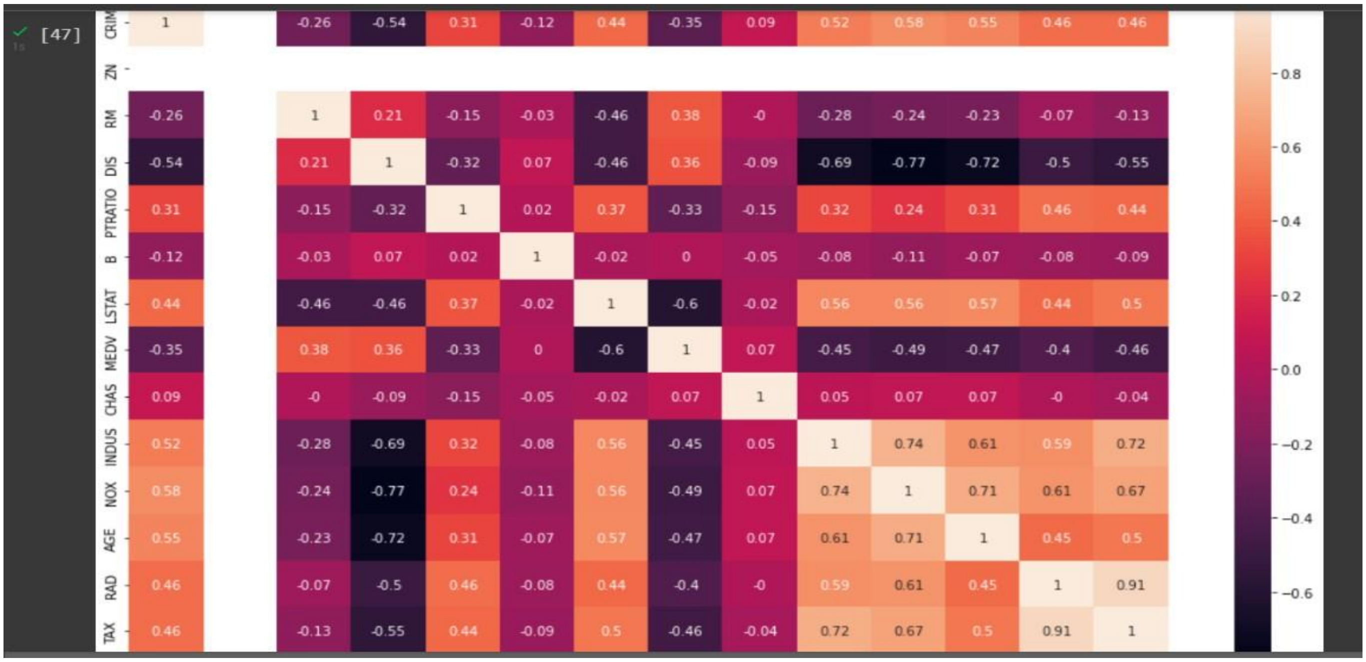


✓ 2s completed at 11:32 PM

```
[47] #heatmap
     fig, ax = plt.subplots(figsize=(17,10))
     correlation_matrix = final_df.corr().round(2)

     sns.heatmap(data=correlation_matrix, annot=True)
```

#Heatmap




```
[48] # correlation between these variables
     print("PEARSON CORRELATION")
     print(final_df.corr(method="pearson"))
     sns.heatmap(final_df.corr(method="pearson"))
     plt.savefig("heatmap_pearson_final.png")
     plt.clf()
     plt.close()
```

[48]

PEARSON CORRELATION

	CRIM	ZN	RM	DIS	PTRATIO	B	LSTAT
CRIM	1.000000	NaN	-0.257687	-0.536575	0.312288	-0.118175	0.438881
ZN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
RM	-0.257687	NaN	1.000000	0.213440	-0.147474	-0.026556	-0.457508
DIS	-0.536575	NaN	0.213440	1.000000	-0.318626	0.066873	-0.459702
PTRATIO	0.312288	NaN	-0.147474	-0.318626	1.000000	0.022006	0.370460
B	-0.118175	NaN	-0.026556	0.066873	0.022006	1.000000	-0.016643
LSTAT	0.438881	NaN	-0.457508	-0.459702	0.370460	-0.016643	1.000000
MEDV	-0.345624	NaN	0.375182	0.360643	-0.325026	0.001044	-0.603648
CHAS	0.094850	NaN	-0.001567	-0.087788	-0.147506	-0.051822	-0.021395
INDUS	0.516031	NaN	-0.284044	-0.690586	0.322969	-0.083767	0.559705
NOX	0.580619	NaN	-0.244543	-0.770106	0.236828	-0.114083	0.557853
AGE	0.548210	NaN	-0.225106	-0.724511	0.311055	-0.070126	0.574629
RAD	0.457619	NaN	-0.074681	-0.502249	0.463409	-0.081552	0.442929
TAX	0.458528	NaN	-0.133313	-0.550279	0.442179	-0.085201	0.497277

	MEDV	CHAS	INDUS	NOX	AGE	RAD	TAX
CRIM	-0.345624	0.094850	0.516031	0.580619	0.548210	0.457619	0.458528
ZN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
RM	0.375182	-0.001567	-0.284044	-0.244543	-0.225106	-0.074681	-0.133313
DIS	0.360643	-0.087788	-0.690586	-0.770106	-0.724511	-0.502249	-0.550279
PTRATIO	-0.325026	-0.147506	0.322969	0.236828	0.311055	0.463409	0.442179
B	0.001044	-0.051822	-0.083767	-0.114083	-0.070126	-0.081552	-0.085201
LSTAT	-0.603648	-0.021395	0.559705	0.557853	0.574629	0.442929	0.497277
MEDV	1.000000	0.074896	-0.446077	-0.488972	-0.470833	-0.401404	-0.460483
CHAS	0.074896	1.000000	0.054172	0.070867	0.073549	-0.003339	-0.035822
INDUS	-0.446077	0.054172	1.000000	0.740965	0.614592	0.593176	0.716062
NOX	-0.488972	0.070867	0.740965	1.000000	0.711461	0.611441	0.668023
AGE	-0.470833	0.073549	0.614592	0.711461	1.000000	0.449989	0.500589
RAD	-0.401404	0.003339	0.593176	0.611441	0.449989	1.000000	0.910228

[49]

```
#scatter plot
plt.figure(figsize=(17,5))

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')
```

#Scatterplot