Experiment - 1

Student Name: Vivek Kumar **UID: 21BCS8129**
Branch: BE-CSE(LEET) **Section/Group: WM-20BCS-616/A**
Semester: 5th **Date of Performance: 16/08/2022**
Subject Name: Machine Learning Lab **Subject Code: 20CSP-317**

1. Aim/Overview of the practical:

Implement exploratory data analysis on any data set.

2. Task to be done/ Which logistics used:

We have performed the Data analysis on Student_data.csv and the housing.csv file and perform treatment of Missing Values and Outliers (Preparation of the dataset for analysis by the treatment of the irregularities.

3. Algorithm/Flowchart (For programming-based labs):

4. Steps for experiment/practical/Code:

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb

data = pd.read_csv('/content/drive/MyDrive/Data/Students_data.csv')

data.head()

data.tail()

data["gender"].unique()

data.shape

data.describe()

data.isnull().sum()

data.iloc[:5, 0]
```

```
data.iloc[:5, 1]

data['GPA']

data.groupby('race').agg({'GPA': 'count'})

data.groupby('race').agg({'GPA': 'median'})

data.columns

new_data = data.drop('Calculus1', axis=1)
new_data.head()

"""**Relationship Analysis**"""

corelation = new_data.corr()
sb.heatmap(corelation, xticklabels=corelation.columns, yticklabels=corelation.columns,
annot=True)

sb.pairplot(new_data)

sb.relplot(x='GPA',y='Algebra',hue='race',data=data)

sb.distplot(data['Algebra'])

sb.catplot(x='GPA',kind='box',data=data)

df = pd.read_csv("/content/drive/MyDrive/Data/housing.csv")
df.head()

df.info()

df.head(10)

df.describe()

df.shape

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

# Six variables among them, having 20 missing values in each case
# Approximately, 4 percent observations in each variable has missing values
(20/506)*100
```

```
# getting the column names of the dataset
df.columns

# Detection of outliers among all variable
# %matplotlib inline
plt.subplots(figsize=(17,10))
df.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)

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

cat_mv.isnull().sum()

cat_mv.mode()

# 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()

# dimension (506 Observations and 1 column)
cat_mv.shape

# 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()

"""Each variable has missing values equal to 20 obs"""

num_mv_out.describe()

# 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()
```

```
num_mv_out.shape

# 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

num_mv_noOut.isnull().sum()

"""Each variable has missing values equal to 20 obs"""

# 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()

"""***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 focussed
here because the first category contains outliers. The second and third categories have no
outliers.***
1. num_out = Numerical variables containing outliers (Missing values will be treated with
***median***)--- "CRIM", "ZN", "RM", "DIS", "PTRATIO", "B", "LSTAT", "MEDV"
2. num_noOut = Numerical variables containing "no outliers" (Missing values will be treated
with ***mean***)--- "INDUS", "NOX", "AGE", "RAD", "TAX"
3. cat_out = Categorical variable conatining no outliers --- "CHAS"---- In this variable, the
observation is either 1 or 0
"""

# 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)
df1

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

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

"""Nine variables containing outliers and remain doesn't have outliers

***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)
num_out

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

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

# 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.
"""

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[:, : ]
num_out

# 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")

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

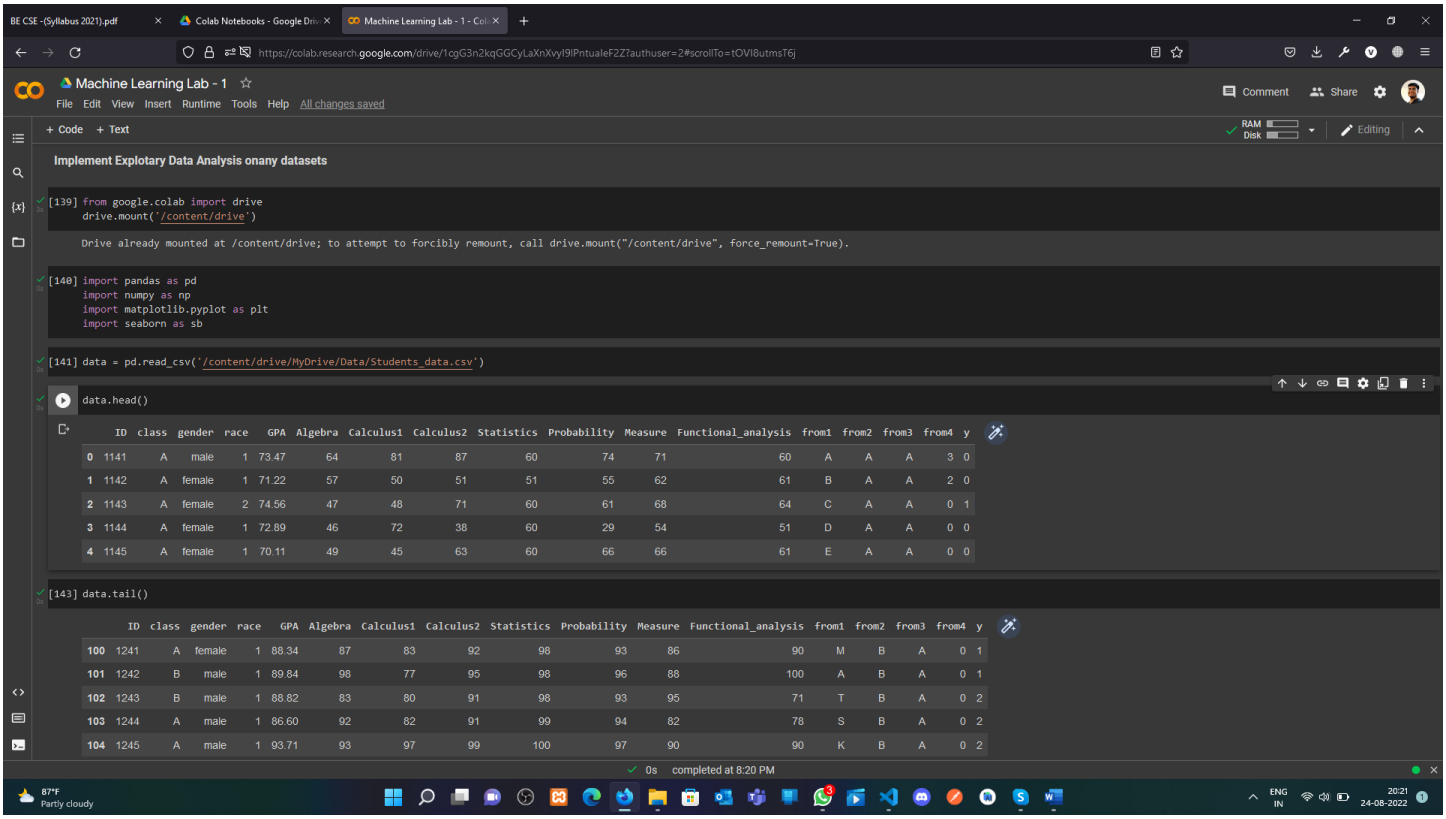
```
# After removing outliers, the following boxplots of each variable from "num_out" show, they have no more outliers
plt.subplots(figsize=(17,10))
num_out.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)

# 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)

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

final_df

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

# Here, Correlation matrix shows:
# the relationship among explanatory variables as well as,
# the relationship between the dependent varibale with each of the explanatory variables
sb.pairplot(final_df)

# the heatmap also shows the same things and interpretations which earlier correlation matrix has been shown
fig, ax = plt.subplots(figsize=(17,10))
correlation_matrix = final_df.corr().round(2)
# annot = True to print the values inside the square
sb.heatmap(data=correlation_matrix, annot=True)

print("PEARSON CORRELATION")
print(final_df.corr(method="pearson"))
sb.heatmap(final_df.corr(method="pearson"))
plt.savefig("heatmap_pearson_final.png")
plt.clf()
plt.close()

#scatter plot to see how these features RAD, RM ,DIS, LSTAT, NOX, AGE, TAX, INDUS vary with Target variable (MEDV)
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')
```

5. Observations/Discussions/ Complexity Analysis:

In this Experiment we have done the Data manipulation in such way where we have found the Unique, count of data, head, Tail, shape, and Descriptive data analysis, iloc, groupby, column, and core relation of the data. Moreover, we have Plotted the graph using seaborn library such as heatmap, pairplot, relplot, distplot and catplot.

6. Result/Output/Writing Summary:



Machine Learning Lab - 1

```
[144] data["gender"].unique()

array(['male', 'female'], dtype=object)

[145] data.shape

(105, 17)

[146] data.describe()
```

	ID	race	GPA	Algebra	Calculus1	Calculus2	Statistics	Probability	Measure	Functional_analysis	from4	y
count	105.000000	105.000000	105.000000	105.000000	105.000000	105.000000	105.000000	105.000000	105.000000	105.000000	105.000000	105.000000
mean	1193.000000	1.790476	82.957048	76.057143	71.961905	78.942857	85.133333	83.876190	80.761905	75.323810	0.504762	0.714286
std	30.454885	1.673867	6.053187	11.722618	12.197039	14.997326	10.269509	10.514363	10.296119	13.003324	0.889293	0.828742
min	1141.000000	1.000000	63.490000	46.000000	38.000000	17.000000	51.000000	29.000000	54.000000	9.000000	0.000000	0.000000
25%	1167.000000	1.000000	79.340000	67.000000	64.000000	71.000000	80.000000	79.000000	74.000000	67.000000	0.000000	0.000000
50%	1193.000000	1.000000	84.110000	76.000000	73.000000	83.000000	87.000000	85.000000	81.000000	76.000000	0.000000	0.000000
75%	1219.000000	1.000000	87.300000	84.000000	80.000000	91.000000	92.000000	92.000000	89.000000	85.000000	0.000000	1.000000
max	1245.000000	7.000000	93.710000	98.000000	100.000000	99.000000	100.000000	97.000000	100.000000	100.000000	3.000000	2.000000

```
[147] data.isnull().sum()

ID                  0
class               0
gender              0
race                0
GPA                 0
Algebra             0
Calculus1           0
Calculus2           0
Statistics          0
Probability         0
Measure             0
Functional_analysis 0
from1               0
from2               0
from3               0
from4               0
y                   0
dtype: int64

[148] data.iloc[:5, 0]

0    1141
1    1142
2    1143
3    1144
4    1145
Name: ID, dtype: int64

[149] data.iloc[:5, 1]

0    A
1    A
2    A
3    A
4    A
Name: class, dtype: object

[150] data['GPA']

0      73.47
1      71.22
2      74.56
3      72.89
4      70.11
       ...
100    88.34
101    89.84
102    88.82
103    86.60
104    93.71
Name: GPA, Length: 105, dtype: float64
```

Machine Learning Lab - 1

File Edit View Insert Runtime Tools Help All changes saved

```
[151] data.groupby('race').agg({'GPA': 'count'})
```

race	GPA
1	81
2	3
3	6
4	4
5	4
6	2
7	5

```
[152] data.groupby('race').agg({'GPA': 'median'})
```

race	GPA
1	84.680
2	74.560
3	83.300
4	78.940
5	75.465
6	83.955
7	81.710

```
[153] data.columns
```

```
Index(['ID', 'class', 'gender', 'race', 'GPA', 'Algebra', 'Calculus1',
       'Calculus2', 'Statistics', 'Probability', 'Measure',
       'Functional_analysis', 'from1', 'from2', 'from3', 'from4', 'y'],
      dtype='object')
```

0s completed at 8:20 PM

Machine Learning Lab - 1

File Edit View Insert Runtime Tools Help All changes saved

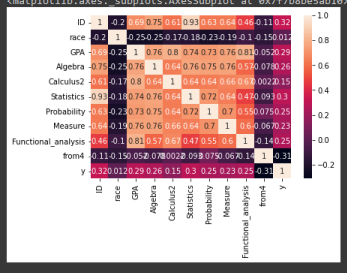
```
[154] new_data = data.drop('Calculus1', axis=1)
      new_data.head()
```

	ID	class	gender	race	GPA	Algebra	Calculus2	Statistics	Probability	Measure	Functional_analysis	from1	from2	from3	from4	y
0	1141	A	male	1	73.47	64	87	60	74	71	60	A	A	A	3	0
1	1142	A	female	1	71.22	57	51	51	55	62	61	B	A	A	2	0
2	1143	A	female	2	74.56	47	71	60	61	68	64	C	A	A	0	1
3	1144	A	female	1	72.89	46	38	60	29	54	51	D	A	A	0	0
4	1145	A	female	1	70.11	49	63	60	66	66	61	E	A	A	0	0

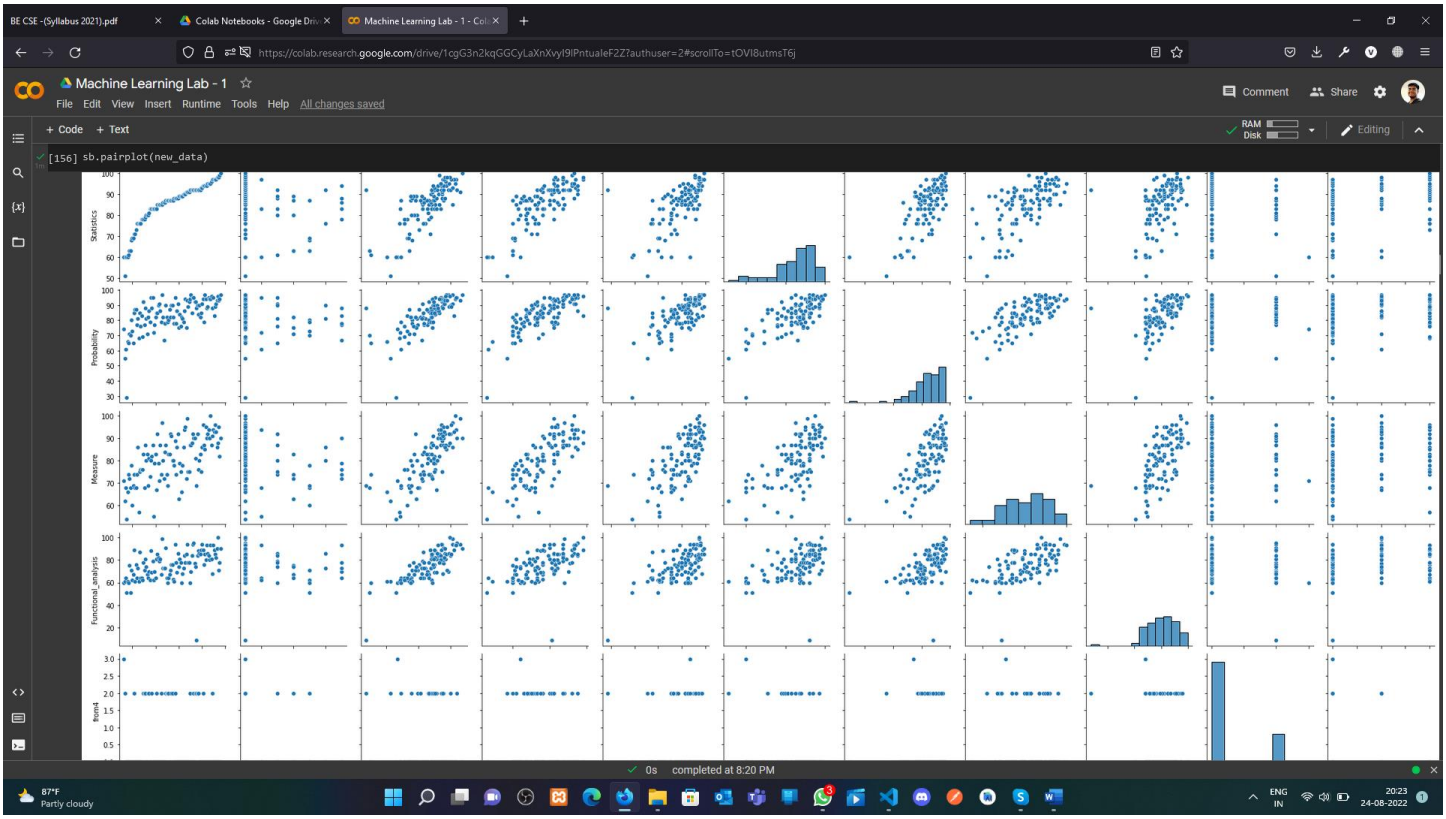
Relationship Analysis

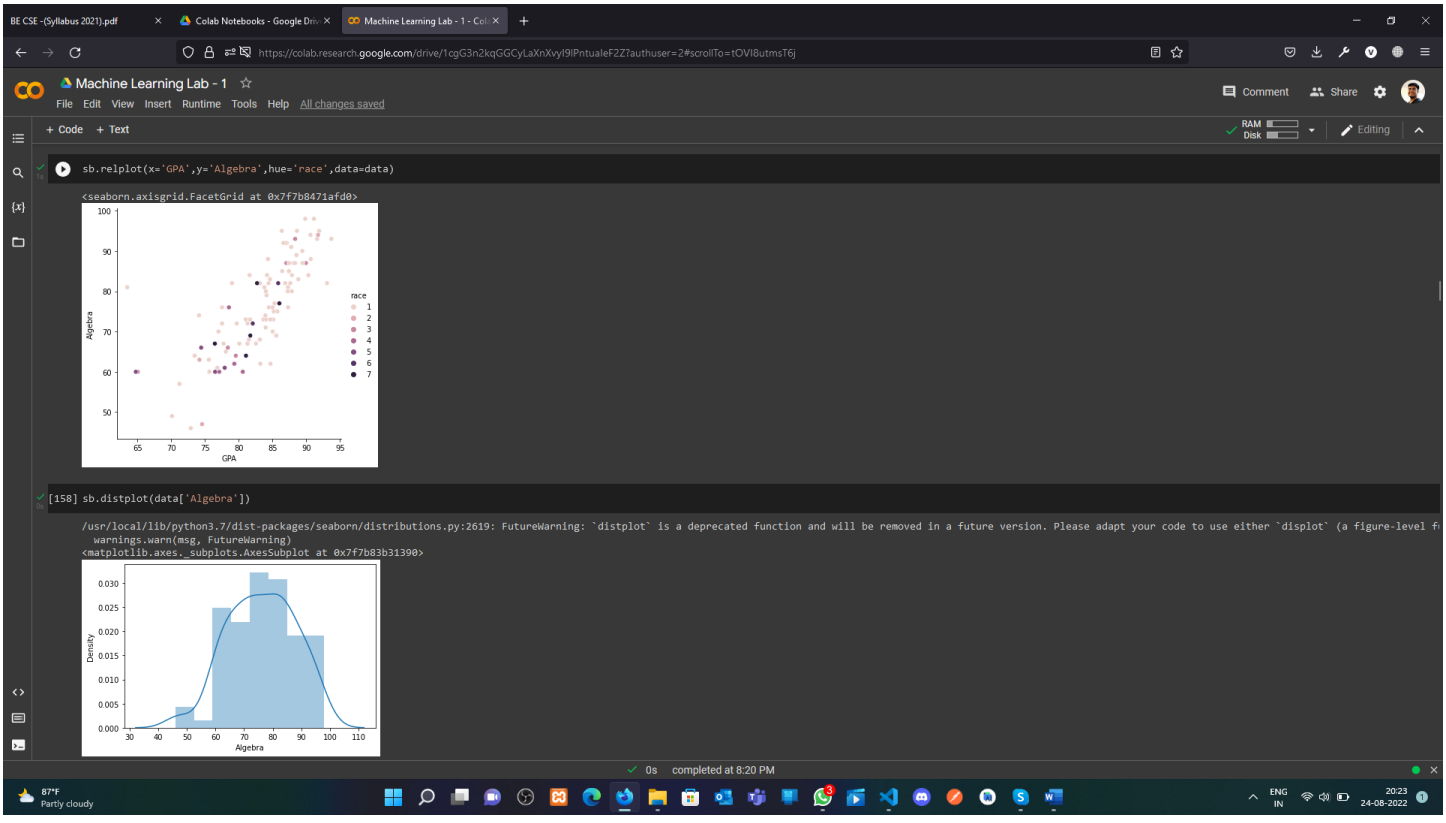
```
[155] corelation = new_data.corr()
      sb.heatmap(corelation, xticklabels=corelation.columns, yticklabels=corelation.columns, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7b8be5ab10>
```



0s completed at 8:20 PM


Machine Learning Lab - 1
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[156] sb.pairplot(new_data)
Statistics
Probability
Measure
Functional_analysis
fitness4
0s completed at 8:20 PM


Machine Learning Lab - 1
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
sb.relplot(x='GPA',y='Algebra',hue='race',data=data)
<seaborn.axisgrid.FacetGrid at 0x7f7b8471afd0>
Algebra
GPA
race
[158] sb.distplot(data['Algebra'])
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level f
warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f7b83b31390>
Density
Algebra
0s completed at 8:20 PM

BE CSE -(Syllabus 2021).pdf | Colab Notebooks - Google Driv | Machine Learning Lab - 1 - Col

https://colab.research.google.com/drive/1cgG3n2kqGGCyLaXnXvyI9lPntualeF2Z?authuser=2#scrollTo=tOVI8utmsT6j

Machine Learning Lab - 1

File Edit View Insert Runtime Tools Help All changes saved

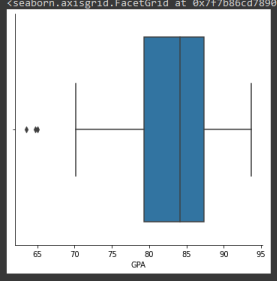
Comment Share

+ Code + Text

RAM Disk Editing

```
[159] sb.catplot(x='GPA',kind='box',data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f7b86cd7890>
```




```
df = pd.read_csv("/content/drive/MyDrive/Data/housing.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

```
[161] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
```

0s completed at 8:20 PM

87°F Partly cloudy

ENG IN 20:24 24-08-2022

BE CSE -(Syllabus 2021).pdf | Colab Notebooks - Google Driv | Machine Learning Lab - 1 - Col

https://colab.research.google.com/drive/1cgG3n2kqGGCyLaXnXvyI9lPntualeF2Z?authuser=2#scrollTo=tOVI8utmsT6j

Machine Learning Lab - 1

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[161] 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
```

```
[162] 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

0s completed at 8:20 PM

87°F Partly cloudy

ENG IN 20:24 24-08-2022

Machine Learning Lab - 1

```
[163] df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
count	486.000000	486.000000	486.000000	486.000000	506.000000	506.000000	486.000000	506.000000	506.000000	506.000000	506.000000	506.000000	486.000000	506.000000
mean	3.611874	11.211934	11.083992	0.069959	0.554695	6.284634	68.518519	3.795043	9.549407	408.237154	18.455534	356.674032	12.715432	22.532806
std	8.720192	23.388876	6.835896	0.255340	0.115878	0.702617	27.999513	2.105710	8.707259	168.537116	2.164946	91.294864	7.155871	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.081900	0.000000	5.190000	0.000000	0.449000	5.885500	45.175000	2.100175	4.000000	279.000000	17.400000	375.377500	7.125000	17.025000
50%	0.253715	0.000000	9.690000	0.000000	0.538000	6.208500	76.800000	3.207450	5.000000	330.000000	19.050000	391.440000	11.430000	21.200000
75%	3.560263	12.500000	18.100000	0.000000	0.624000	6.623500	93.975000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

```
[164] df.shape
```

```
(506, 14)
```

```
[165] # 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
```

```
[166] # Six variables among them, having 20 missing values in each case
```

0s completed at 8:20 PM

87°F Partly cloudy — ENG IN — 20:25 24-08-2022

Machine Learning Lab - 1

```
[166] (20/506)*100
```

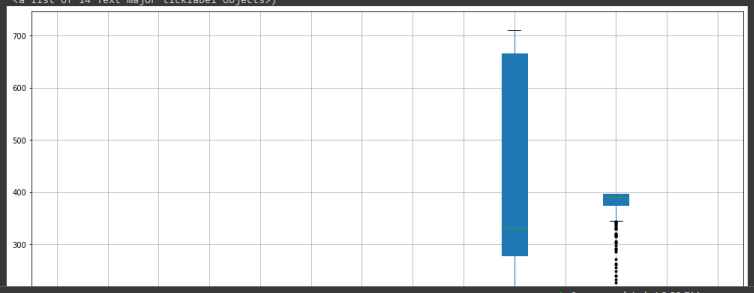
```
3.9525691699604746
```

```
[167] # 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')
```

```
# Detection of outliers among all variable
# %matplotlib inline
plt.subplots(figsize=(17,10))
df.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different leng
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 <a list of 14 Text major ticklabel objects>)
```



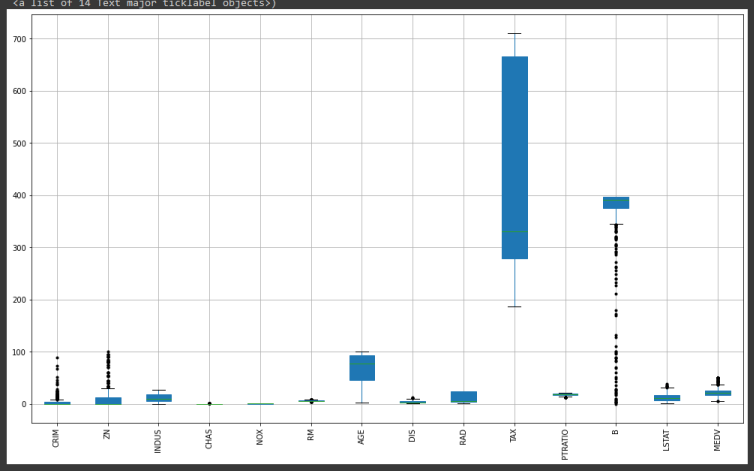
0s completed at 8:20 PM

87°F Partly cloudy — ENG IN — 20:26 24-08-2022

Machine Learning Lab - 1

```
# Detection of outliers among all variable
# %matplotlib inline
plt.subplots(figsize=(17,10))
df.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different leng
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 <a list of 14 Text major ticklabel objects>)
```



0s completed at 8:20 PM

```
[169] # 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

506 rows × 1 columns

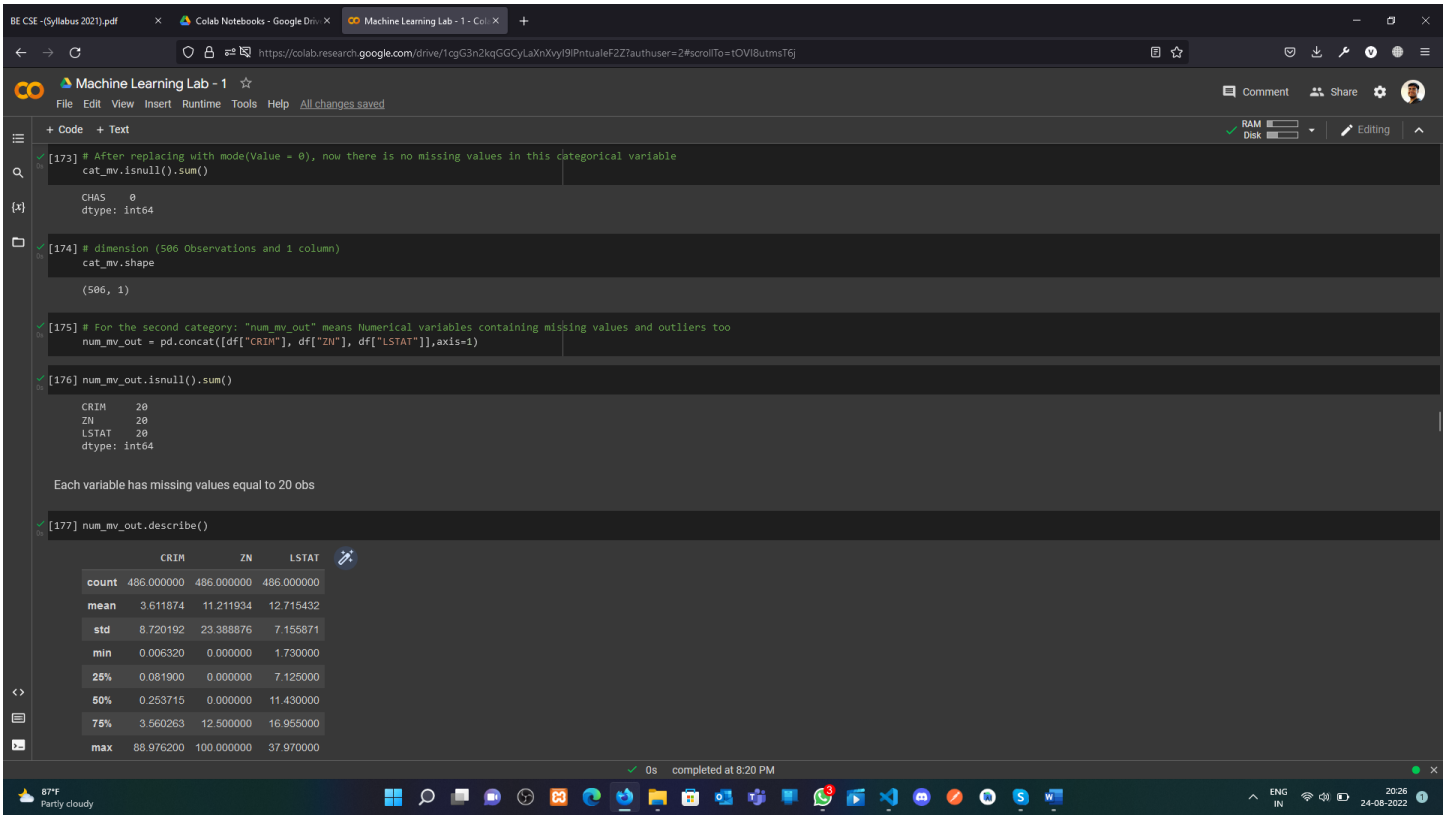
```
[170] cat_mv.isnull().sum()

CHAS    20
dtype: int64
```

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

	CHAS
0	0.0

```
[172] # 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)
```

0s completed at 8:20 PM



Machine Learning Lab - 1

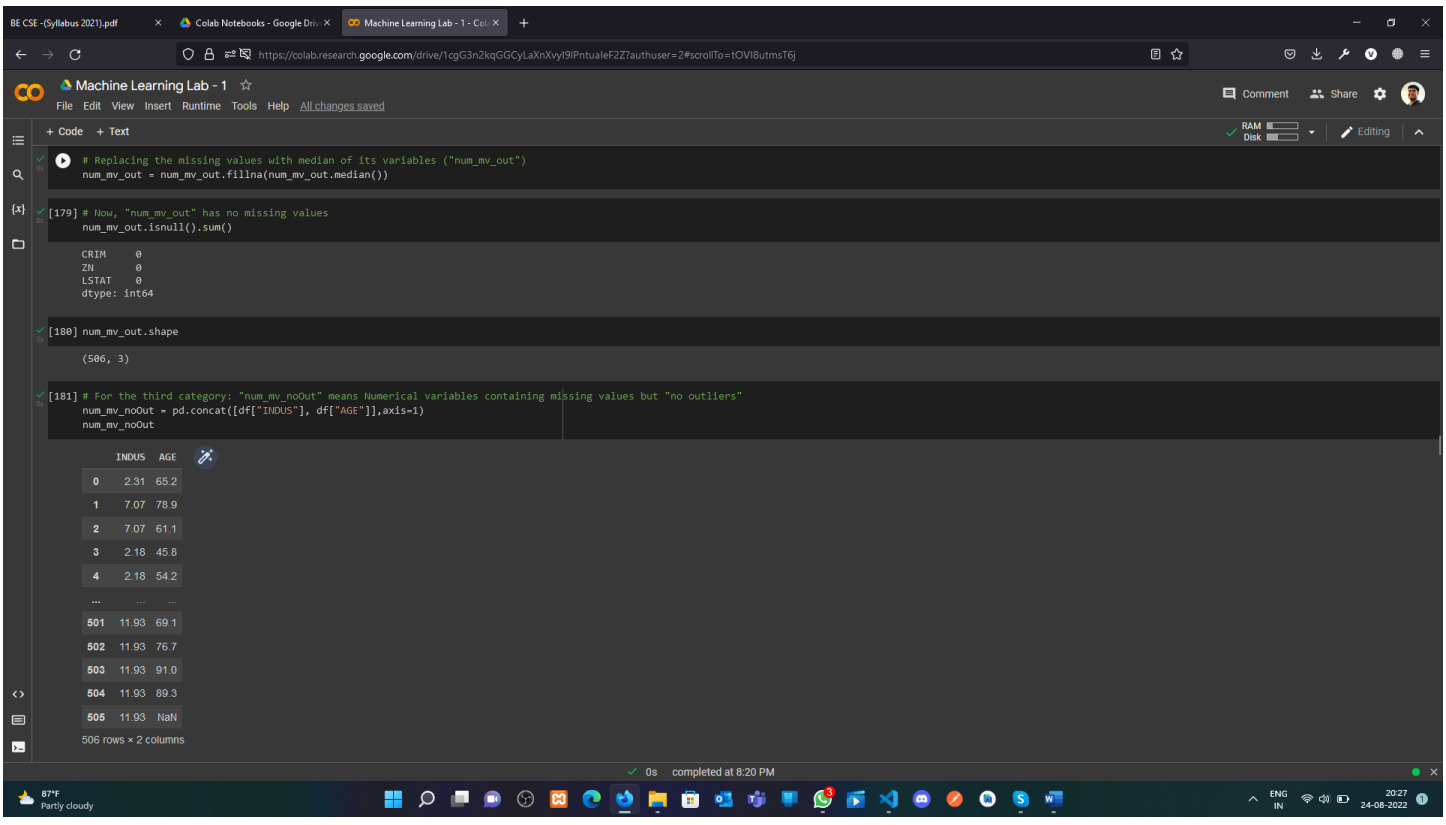
```
[173] # After replacing with mode(Value = 0), now there is no missing values in this categorical variable
      cat_mv.isnull().sum()

      CHAS    0
      dtype: int64

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

      (506, 1)

[175] # 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)

[176] num_mv_out.isnull().sum()

      CRIM     20
      ZN       20
      LSTAT    20
      dtype: int64
```

Each variable has missing values equal to 20 obs

```
[177] num_mv_out.describe()
```

	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

0s completed at 8:20 PM



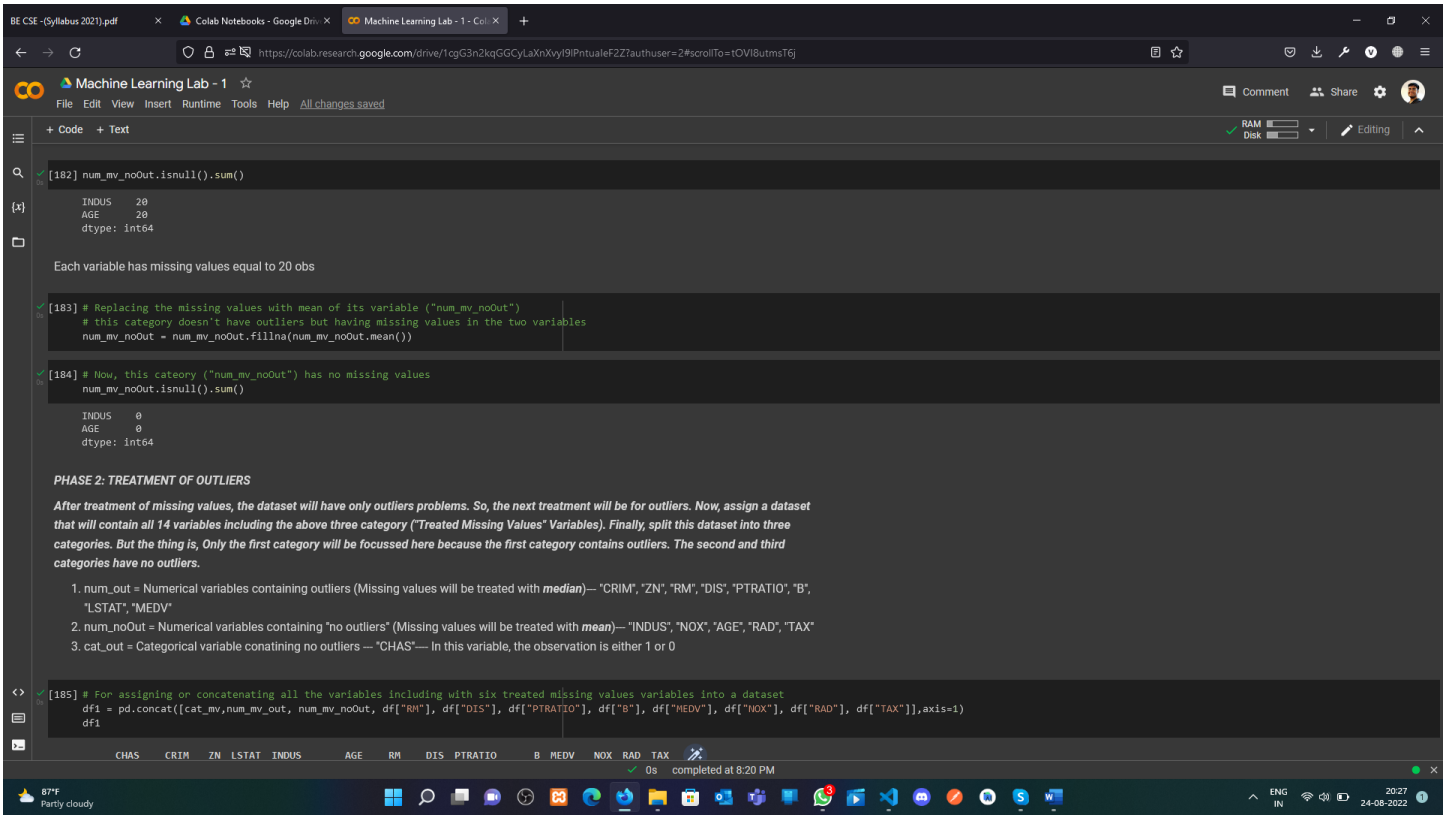
Machine Learning Lab - 1

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

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

      CRIM     0
      ZN       0
      LSTAT    0
      dtype: int64

[180] num_mv_out.shape

      (506, 3)

[181] # 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

0s completed at 8:20 PM



Machine Learning Lab - 1

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

INDUS    20
AGE      20
dtype: int64
```

Each variable has missing values equal to 20 obs

```
[183] # 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())
```

```
[184] # 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 focussed here because the first category contains outliers. The second and third categories have no outliers.

1. num_out = Numerical variables containing outliers (Missing values will be treated with ***median***)— "CRIM", "ZN", "RM", "DIS", "PTRATIO", "B", "LSTAT", "MEDV"
2. num_noOut = Numerical variables containing "no outliers" (Missing values will be treated with ***mean***)— "INDUS", "NOX", "AGE", "RAD", "TAX"
3. cat_out = Categorical variable conatining no outliers — "CHAS"— In this variable, the observation is either 1 or 0

```
[185] # 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)
      df1
```

	CHAS	CRIM	ZN	LSTAT	INDUS	AGE	RM	DIS	PTRATIO	B	MEDV	NOX	RAD	TAX
0	0.0	0.00632	18.0	4.98	2.31	65.200000	6.575	4.0900	15.3	396.90	24.0	0.538	1	296
1	0.0	0.02731	0.0	9.14	7.07	78.900000	6.421	4.9671	17.8	396.90	21.6	0.469	2	242
2	0.0	0.02729	0.0	4.03	7.07	61.100000	7.185	4.9671	17.8	392.83	34.7	0.469	2	242
3	0.0	0.03237	0.0	2.94	2.18	45.800000	6.998	6.0622	18.7	394.63	33.4	0.458	3	222
4	0.0	0.06905	0.0	11.43	2.18	54.200000	7.147	6.0622	18.7	396.90	36.2	0.458	3	222
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.0	0.06263	0.0	11.43	11.93	69.100000	6.593	2.4786	21.0	391.99	22.4	0.573	1	273
502	0.0	0.04527	0.0	9.08	11.93	76.700000	6.120	2.2875	21.0	396.90	20.6	0.573	1	273
503	0.0	0.06076	0.0	5.64	11.93	91.000000	6.976	2.1675	21.0	396.90	23.9	0.573	1	273
504	0.0	0.10959	0.0	6.48	11.93	89.300000	6.794	2.3889	21.0	393.45	22.0	0.573	1	273
505	0.0	0.04741	0.0	7.88	11.93	68.518519	6.030	2.5050	21.0	396.90	11.9	0.573	1	273

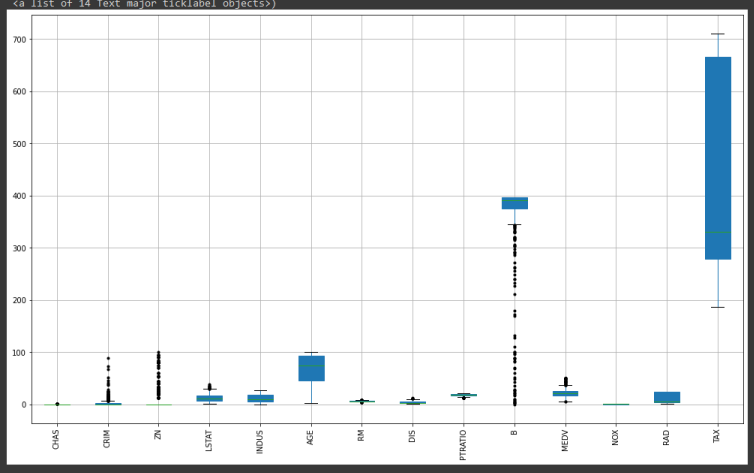
506 rows × 14 columns

```
[186] # 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
```

Machine Learning Lab - 1

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

```
(array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 <a list of 14 Text major ticklabel objects>)
```



Nine variables containing outliers and remain doesn't have outliers

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)
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

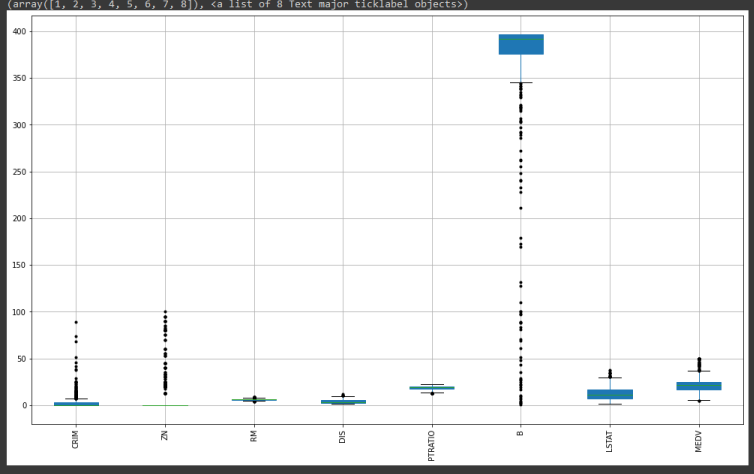
```
# Detecting outliers in "cat_out"
plt.subplots(figsize=(17,10))
num_out.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

```
(array([1, 2, 3, 4, 5, 6, 7, 8]), <a list of 8 Text major ticklabel objects>)
```

Machine Learning Lab - 1

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

```
(array([1, 2, 3, 4, 5, 6, 7, 8]), <a list of 8 Text major ticklabel objects>)
```

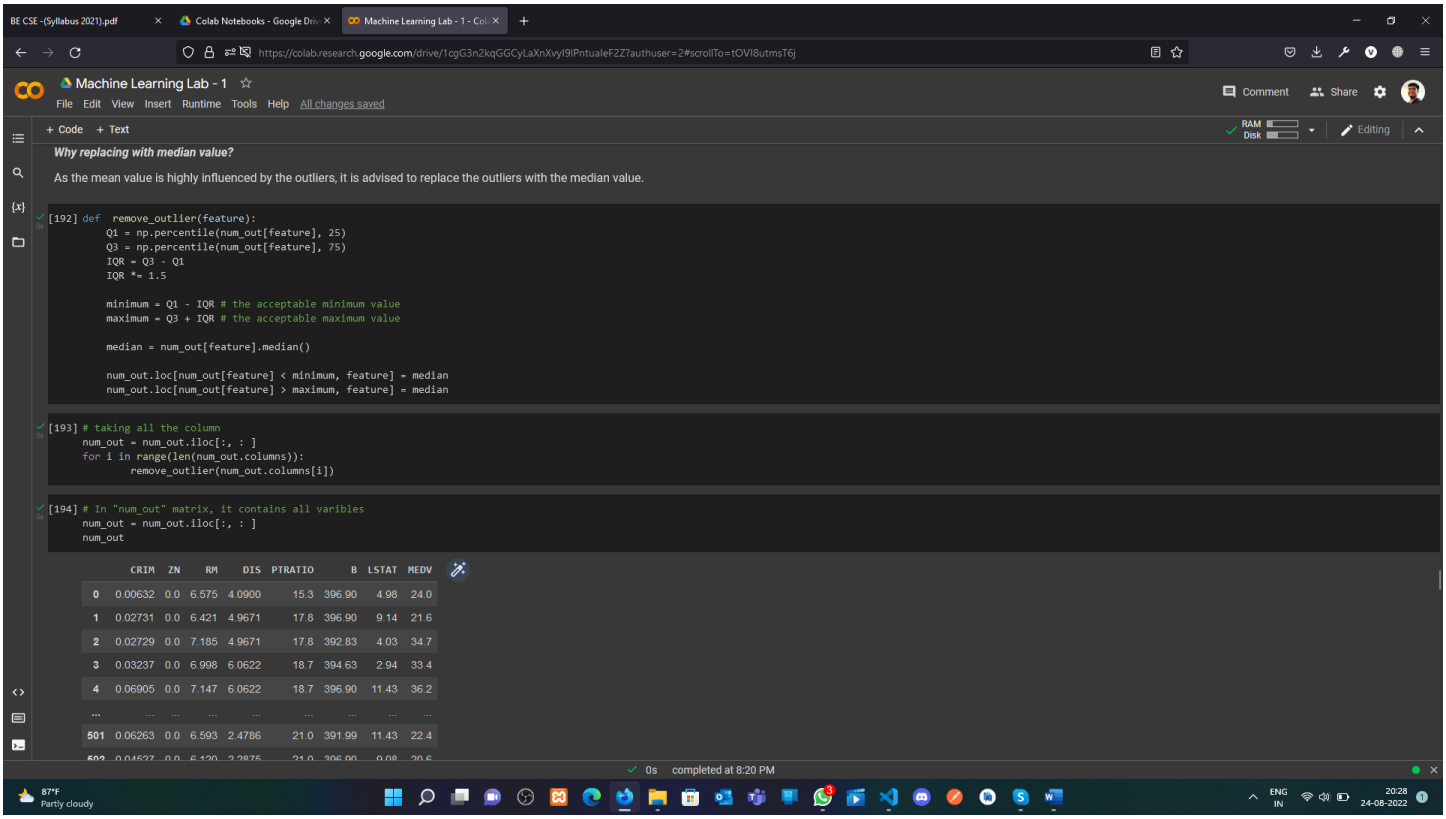



```
[190] # Getting the basic statistical summary of those variables containing outliers
num_out.describe()
```

	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

```
[191] # 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.


Machine Learning Lab - 1
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
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.
[192] 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
[193] # taking all the column
num_out = num_out.iloc[:, :]
for i in range(len(num_out.columns)):
remove_outlier(num_out.columns[i])
[194] # 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
0s completed at 8:20 PM
87°F Partly cloudy
ENG IN
20:28 24-08-2022

Machine Learning Lab - 1

	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

```
# 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
```

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

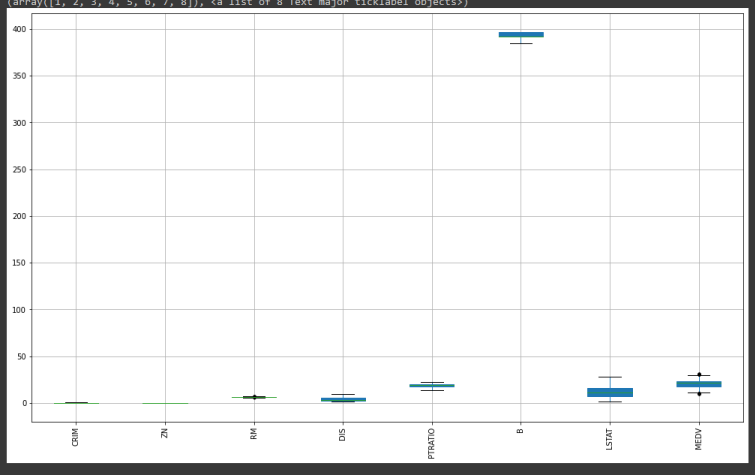
```
# After removing outliers, the following boxplots of each variable from "num_out" show, they have no more outliers
plt.subplots(figsize=(17,10))
num_out.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

0s completed at 8:20 PM

Machine Learning Lab - 1

```
# 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)
```

```
(array([1, 2, 3, 4, 5, 6, 7, 8]), <a list of 8 Text major ticklabel objects>)
```




```
# 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)
```

0s completed at 8:20 PM

Machine Learning Lab - 1

After treatment of missing values as well as outliers

The dataset is now ready for further analysis

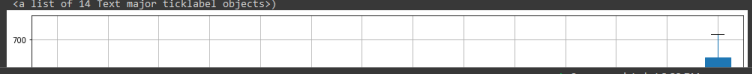
[199] final_df

	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

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

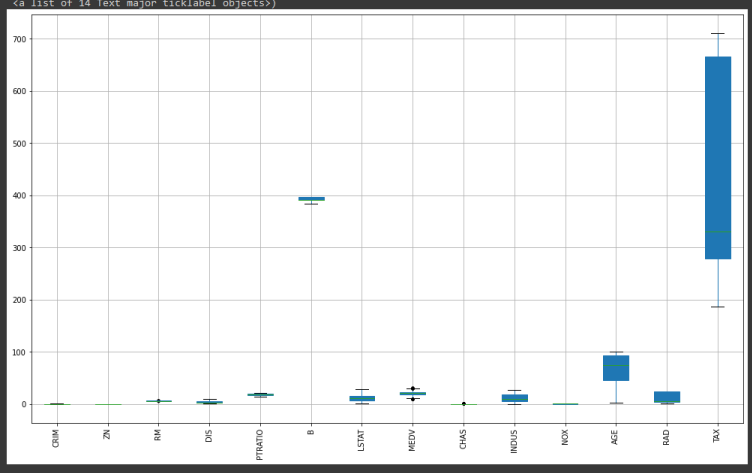
(array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]),
<a list of 14 Text major ticklabel objects>)



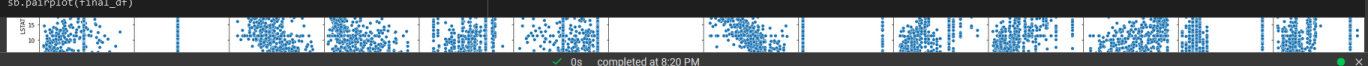
0s completed at 8:20 PM

Machine Learning Lab - 1

[200] (array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]),
<a list of 14 Text major ticklabel objects>)



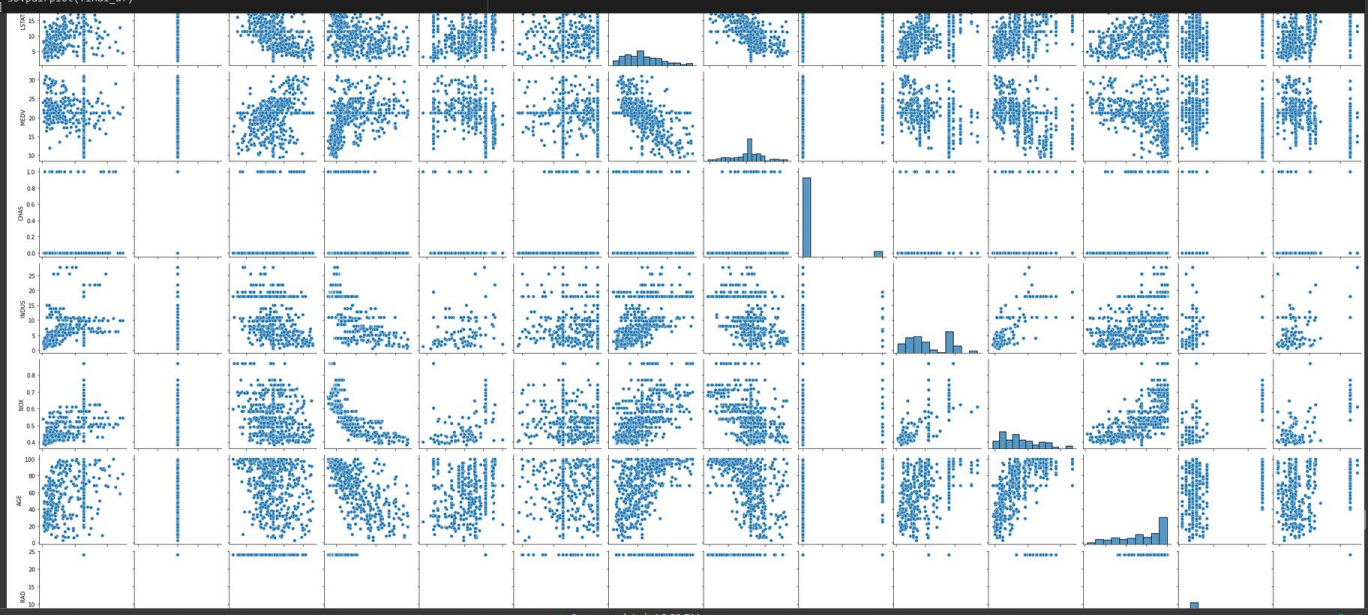
[201] # Here, Correlation matrix shows:
# the relationship among explanatory variables as well as,
# the relationship between the dependent varibale with each of the explanatory variables
sb.pairplot(final_df)



0s completed at 8:20 PM

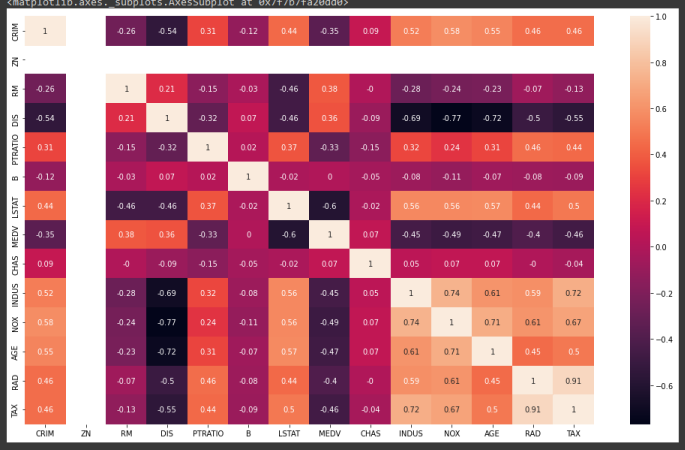
Machine Learning Lab - 1

```
[201] sb.pairplot(final_df)
```



```
[202] # the heatmap also shows the same things and interpretations which earlier correlation matrix has been shown
      fig, ax = plt.subplots(figsize=(17,10))
      correlation_matrix = final_df.corr().round(2)
      # annot = True to print the values inside the square
      sb.heatmap(data=correlation_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7b7fa20dd0>
```



Machine Learning Lab - 1

```
[203] print("PEARSON CORRELATION")
      print(final_df.corr(method="pearson"))
      sb.heatmap(final_df.corr(method="pearson"))
      plt.savefig("heatmap_pearson_final.png")
      plt.clf()
      plt.close()
```

```
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 -0.003767  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 -0.003767 -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  0.074096 -0.446077 -0.488972 -0.470833 -0.401404 -0.460483
CHAS     0.074096  1.000000  0.054172  0.070867  0.073549 -0.003339 -0.035822
INDUS   -0.446077  0.054172  1.000000  0.740065  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
TAX     -0.460483 -0.035822  0.716062  0.668023  0.500589  0.910228  1.000000
```

```
[204] #scatter plot to see how these features RAD, RM ,DIS, LSTAT, NOX, AGE, TAX, INDUS vary with Target variable (MEDV)
      plt.figure(figsize=(17,5))
```

```
<Figure size 1224x360 with 0 Axes>
<Figure size 1224x360 with 0 Axes>
```

0s completed at 8:20 PM

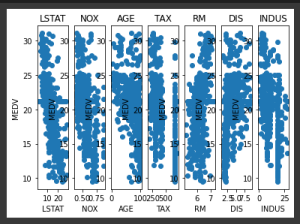
87°F Partly cloudy

ENG IN 20:30 24-08-2022

Machine Learning Lab - 1

```
[205] 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')
```



0s completed at 8:20 PM

87°F Partly cloudy

ENG IN 20:30 24-08-2022

Learning outcomes (What I have learnt):

1. Data manipulation using padas library

2. Data plotting using seaborn library

3. Missing values rectification

4. Outliers treatments

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.			