In [177…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [178…
```
df = pd.read_csv('housing_data.csv')
df
```

Out[178…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90
…	…	…	…	…	…	…	…	…	…	…	…	…
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90

506 rows × 14 columns

In [179…
```
df.head()
```

Out[179…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	I
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	

In [180…
```
df.shape
```

Out[180…
```
(506, 14)
```

In [181…
```
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
```

In [182…
```
df.isnull().sum()
```

Out[182…
```
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
```

In [183…
```
nc=['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX', 'PTRATIO','B','L
for col in nc:
    df[col].fillna(df[col].median(), inplace =True)
df
```

Out[183…

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90
…	…	…	…	…	…	…	…	…	…	…	…	…
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	76.8	2.5050	1	273	21.0	396.90

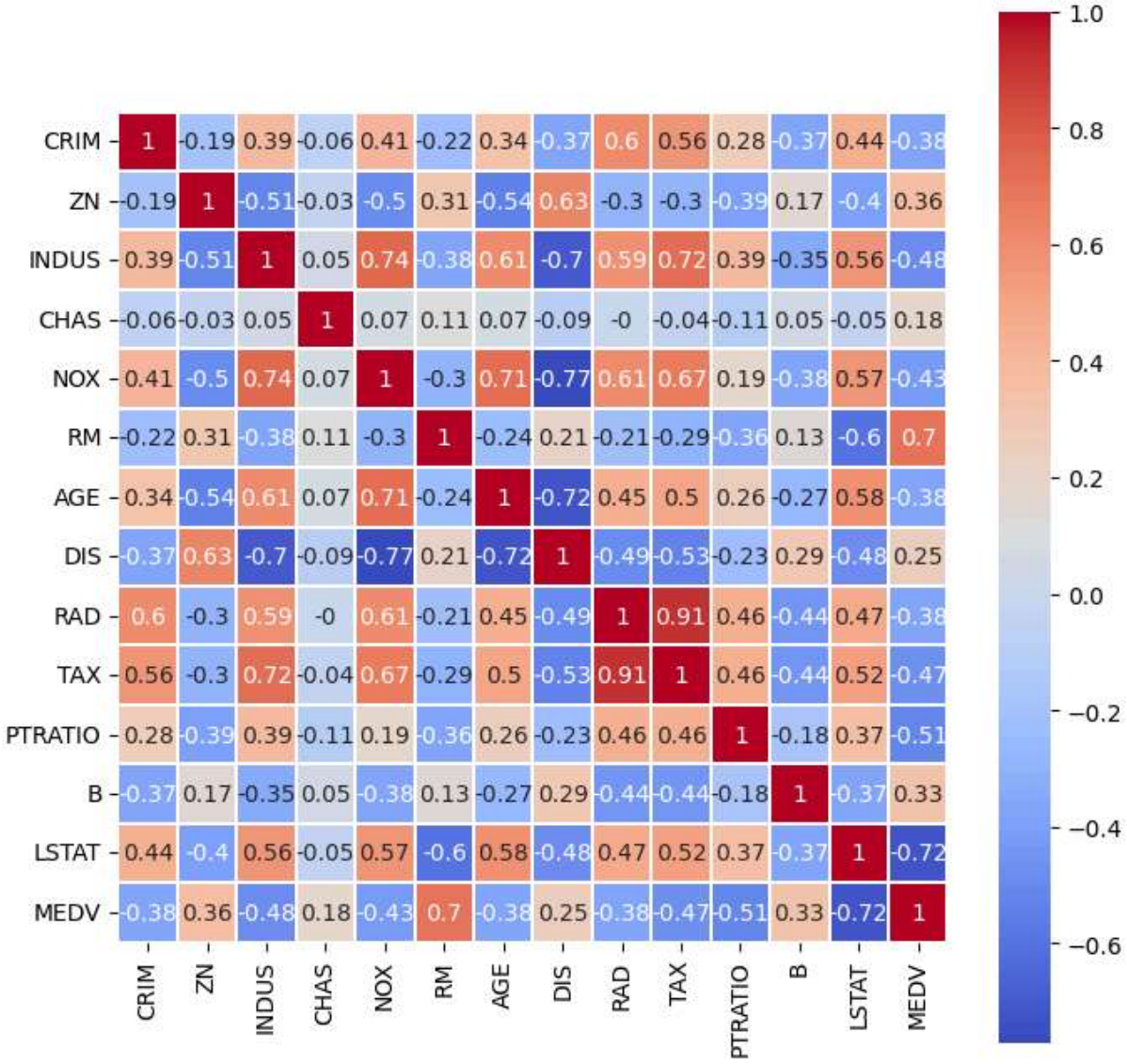
506 rows × 14 columns

In [184…
```
df.isnull().sum()
```

Out[184…
```
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
```

In [185…
```
plt.figure(figsize=(8,8))
sns.heatmap(data=df.corr().round(2),annot=True,cmap='coolwarm',linewidths=0.2,squar
```

Out[185…
```
<Axes: >
```



In [186…

```
df1=df[['RM','TAX','PTRATIO','LSTAT','MEDV']]
df1
```

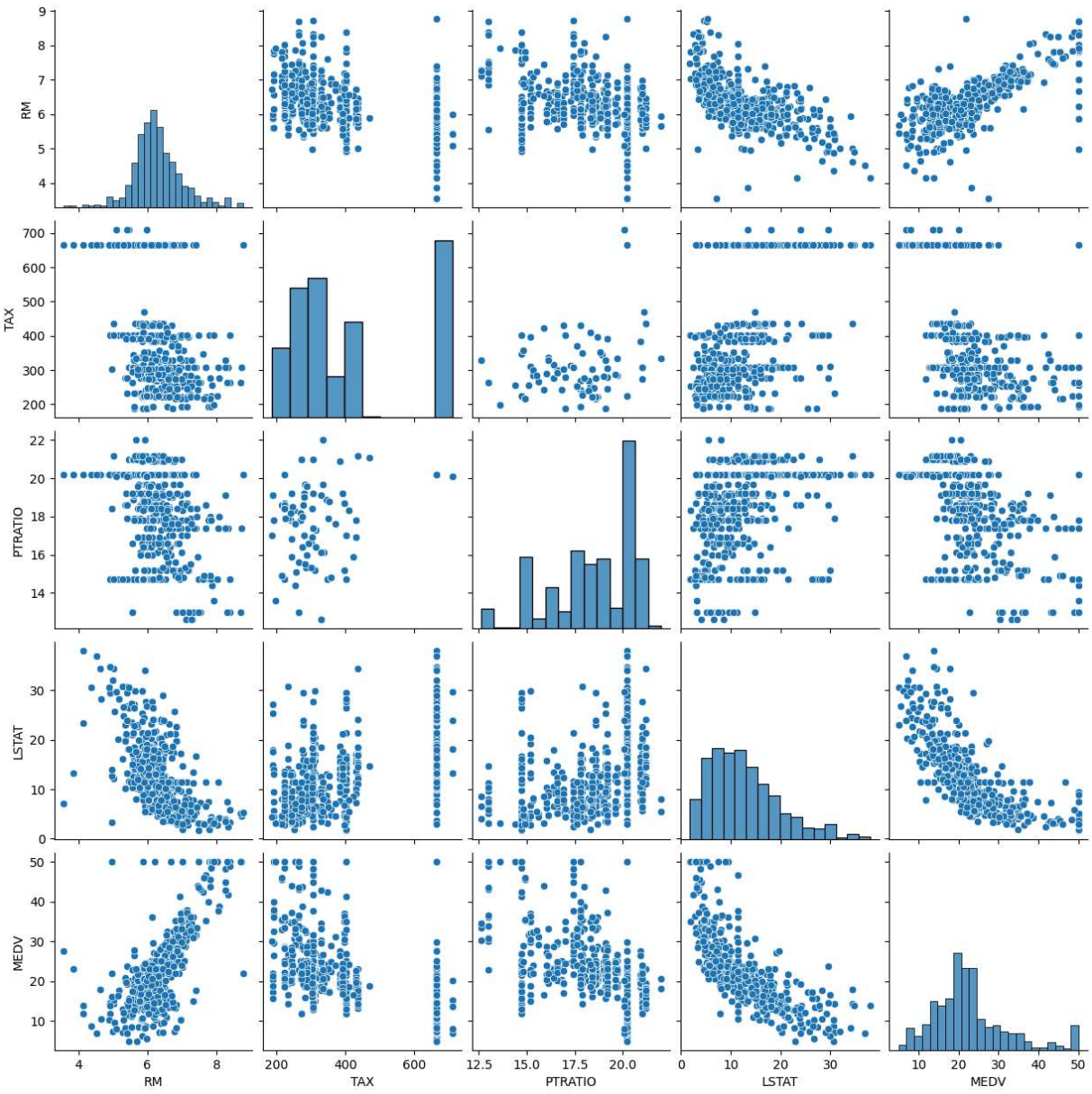
Out[186…

	RM	TAX	PTRATIO	LSTAT	MEDV
0	6.575	296	15.3	4.98	24.0
1	6.421	242	17.8	9.14	21.6
2	7.185	242	17.8	4.03	34.7
3	6.998	222	18.7	2.94	33.4
4	7.147	222	18.7	11.43	36.2
…	…	…	…	…	…
501	6.593	273	21.0	11.43	22.4
502	6.120	273	21.0	9.08	20.6
503	6.976	273	21.0	5.64	23.9
504	6.794	273	21.0	6.48	22.0
505	6.030	273	21.0	7.88	11.9

506 rows × 5 columns

In [187…
```
sns.pairplot(data=df1)
```

Out[187…
```
<seaborn.axisgrid.PairGrid at 0x2b4bb727950>
```



In [188…

```
D = df1.describe().round(2)
D
```

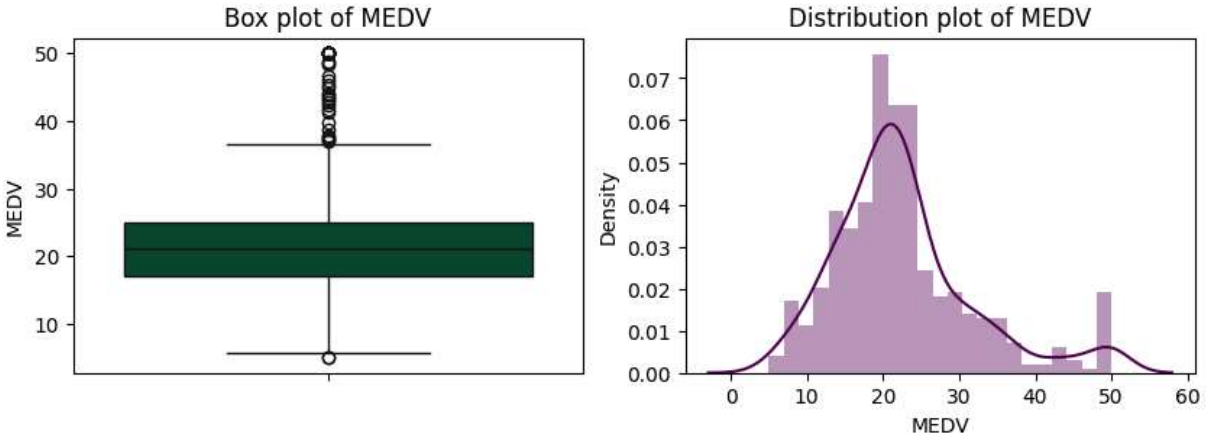
Out[188…

	RM	TAX	PTRATIO	LSTAT	MEDV
count	506.00	506.00	506.00	506.00	506.00
mean	6.28	408.24	18.46	12.66	22.53
std	0.70	168.54	2.16	7.02	9.20
min	3.56	187.00	12.60	1.73	5.00
25%	5.89	279.00	17.40	7.23	17.02
50%	6.21	330.00	19.05	11.43	21.20
75%	6.62	666.00	20.20	16.57	25.00
max	8.78	711.00	22.00	37.97	50.00

In [189…
```
df1.shape
```

Out[189… (506, 5)

In [190…
```
plt.figure(figsize=(10,3))

plt.subplot(1,2,1)
sns.boxplot(df1.MEDV,color='#005030')
plt.title('Box plot of MEDV')

# plt.figure(figsize=(20,3))
plt.subplot(1,2,2)
sns.distplot(a=df1.MEDV,color='#500050')
plt.title('Distribution plot of MEDV')

plt.show()
```



In [191…
```
MEDV_Q3=D['MEDV']['75%']
MEDV_Q1=D['MEDV']['25%']
MEDV_IQR=MEDV_Q3-MEDV_Q1
MEDV_LV=MEDV_Q1-1.5*MEDV_IQR
MEDV_UV=MEDV_Q3+1.5*MEDV_IQR
df1[df1['MEDV']<MEDV_LV]
```

Out[191…

	RM	TAX	PTRATIO	LSTAT	MEDV
398	5.453	666	20.2	30.59	5.0
405	5.683	666	20.2	22.98	5.0

In [192…

```
df1[df1['MEDV']>MEDV_UV]
```

Out[192...

	RM	TAX	PTRATIO	LSTAT	MEDV
97	8.069	276	18.0	4.21	38.7
98	7.820	276	18.0	3.57	43.8
157	6.943	403	14.7	4.59	41.3
161	7.489	403	14.7	1.73	50.0
162	7.802	403	14.7	1.92	50.0
163	8.375	403	14.7	3.32	50.0
166	7.929	403	14.7	3.70	50.0
179	6.980	193	17.8	5.04	37.2
180	7.765	193	17.8	7.56	39.8
182	7.155	193	17.8	4.82	37.9
186	7.831	193	17.8	4.45	50.0
190	6.951	398	15.2	5.10	37.0
195	7.875	255	14.4	2.97	50.0
202	7.610	348	14.7	3.11	42.3
203	7.853	224	14.7	3.81	48.5
204	8.034	224	14.7	2.88	50.0
224	8.266	307	17.4	4.14	44.8
225	8.725	307	17.4	4.63	50.0
226	8.040	307	17.4	11.43	37.6
228	7.686	307	17.4	11.43	46.7
232	8.337	307	17.4	2.47	41.7
233	8.247	307	17.4	3.95	48.3
253	8.259	330	19.1	3.54	42.8
256	7.454	244	15.9	3.11	44.0
257	8.704	264	13.0	5.12	50.0
261	7.520	264	13.0	7.26	43.1
262	8.398	264	13.0	5.91	48.8
267	8.297	264	13.0	7.44	50.0
268	7.470	264	13.0	3.16	43.5
280	7.820	216	14.9	3.76	45.4

	RM	TAX	PTRATIO	LSTAT	MEDV
282	7.645	216	14.9	3.01	46.0
283	7.923	198	13.6	3.16	50.0
291	7.148	245	19.2	3.56	37.3
368	4.970	666	20.2	3.26	50.0
369	6.683	666	20.2	3.73	50.0
370	7.016	666	20.2	2.96	50.0
371	6.216	666	20.2	9.53	50.0
372	5.875	666	20.2	8.88	50.0

In [193…
```
df2 = df1[~(df1['MEDV']==50)]
df2
```

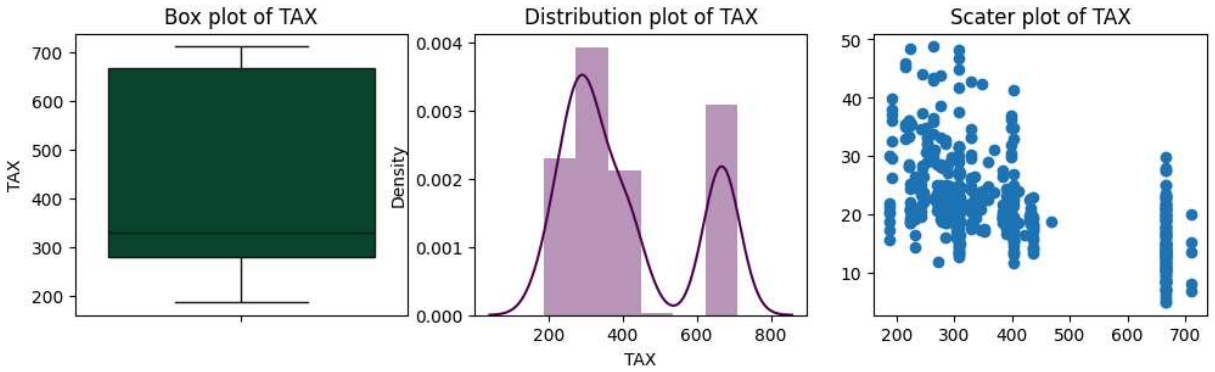
Out[193…

	RM	TAX	PTRATIO	LSTAT	MEDV
0	6.575	296	15.3	4.98	24.0
1	6.421	242	17.8	9.14	21.6
2	7.185	242	17.8	4.03	34.7
3	6.998	222	18.7	2.94	33.4
4	7.147	222	18.7	11.43	36.2
…	…	…	…	…	…
501	6.593	273	21.0	11.43	22.4
502	6.120	273	21.0	9.08	20.6
503	6.976	273	21.0	5.64	23.9
504	6.794	273	21.0	6.48	22.0
505	6.030	273	21.0	7.88	11.9

490 rows × 5 columns

In [194…
```
# For tax
plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.boxplot(df2.TAX,color='#005030')
plt.title('Box plot of TAX')

plt.subplot(1,3,2)
sns.distplot(a=df2.TAX,color='#500050')
plt.title('Distribution plot of TAX')
```

```
plt.subplot(1,3,3)
plt.scatter(y=df2.MEDV, x=df2.TAX)
plt.title('Scater plot of TAX')

plt.show()
```



In [195…
```
df2[df1['TAX']>600]
```

Out[195…

	RM	TAX	PTRATIO	LSTAT	MEDV
356	6.212	666	20.2	17.60	17.8
357	6.395	666	20.2	13.27	21.7
358	6.127	666	20.2	11.48	22.7
359	6.112	666	20.2	12.67	22.6
360	6.398	666	20.2	7.79	25.0
...	...	...	...	...	...
488	5.454	711	20.1	18.06	15.2
489	5.414	711	20.1	23.97	7.0
490	5.093	711	20.1	29.68	8.1
491	5.983	711	20.1	18.07	13.6
492	5.983	711	20.1	13.35	20.1

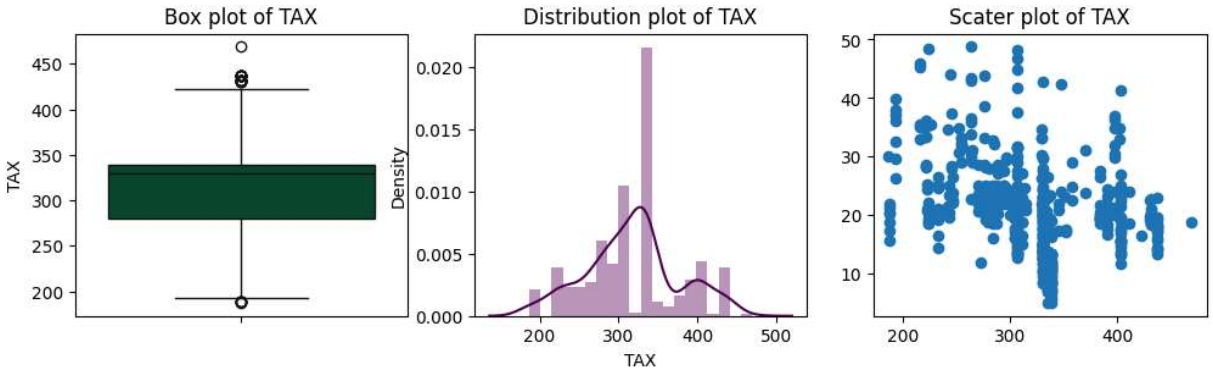
132 rows × 5 columns

In [196…
```
TAX_10 = df2[(df2['TAX']<600) & (df2['LSTAT']>=0) & (df2['LSTAT']<10)]['TAX'].mean(
TAX_20 = df2[(df2['TAX']<600) & (df2['LSTAT']>=10) & (df2['LSTAT']<20)]['TAX'].mean
TAX_30 = df2[(df2['TAX']<600) & (df2['LSTAT']>=20) & (df2['LSTAT']<30)]['TAX'].mean
TAX_40 = df2[(df2['TAX']<600) & (df2['LSTAT']>=30)]['TAX'].mean()

indexes = list(df2.index)
for i in indexes:
    if(df2['TAX'][i]>600):
        if(0 <= df2['LSTAT'][i] < 10):
            df2.at[i,'TAX'] = TAX_10
        elif(10 <= df2['LSTAT'][i] < 20):
```

```
            df2.at[i,'TAX'] = TAX_20
        elif(20 <= df2['LSTAT'][i] < 30):
            df2.at[i,'TAX'] = TAX_30
        elif(df2['LSTAT'][i] >= 30):
            df2.at[i,'TAX'] = TAX_40

df2[df2['TAX']>600]['TAX'].count()
```

Out[196… 0

In [197…
```
plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.boxplot(df2.TAX,color='#005030')
plt.title('Box plot of TAX')

plt.subplot(1,3,2)
sns.distplot(a=df2.TAX,color='#500050')
plt.title('Distribution plot of TAX')

plt.subplot(1,3,3)
plt.scatter(y=df2.MEDV, x=df2.TAX)
plt.title('Scater plot of TAX')

plt.show()
```



In [198…
```
D1 = df2.describe().round(2)
D1
```

Out[198…

	RM	TAX	PTRATIO	LSTAT	MEDV
count	490.00	490.00	490.00	490.00	490.00
mean	6.25	317.54	18.52	12.94	21.64
std	0.65	58.32	2.11	6.95	7.87
min	3.56	187.00	12.60	1.98	5.00
25%	5.88	280.25	17.40	7.54	16.70
50%	6.18	330.00	19.10	11.43	20.90
75%	6.58	338.64	20.20	16.93	24.68
max	8.78	469.00	22.00	37.97	48.80

In [199…

```
TAX_Q3=D1['TAX']['75%']
TAX_Q1=D1['TAX']['25%']
TAX_IQR=TAX_Q3-TAX_Q1
TAX_LV=TAX_Q1-1.5*TAX_IQR
TAX_UV=TAX_Q3+1.5*TAX_IQR
```

In [200…

```
df2[df2['TAX']<TAX_LV]
```

Out[200…

	RM	TAX	PTRATIO	LSTAT	MEDV
120	5.870	188.0	19.1	14.37	22.0
121	6.004	188.0	19.1	14.27	20.3
122	5.961	188.0	19.1	17.93	20.5
123	5.856	188.0	19.1	25.41	17.3
124	5.879	188.0	19.1	17.58	18.8
125	5.986	188.0	19.1	14.81	21.4
126	5.613	188.0	19.1	27.26	15.7
353	6.728	187.0	17.0	4.50	30.1

In [201…

```
df2.shape
```

Out[201… (490, 5)

In [202…

```
df2 = df2.drop(index=353, axis=0)
df2.shape
```

Out[202… (489, 5)

In [203…

```
df2[df2['TAX']>TAX_UV]
```

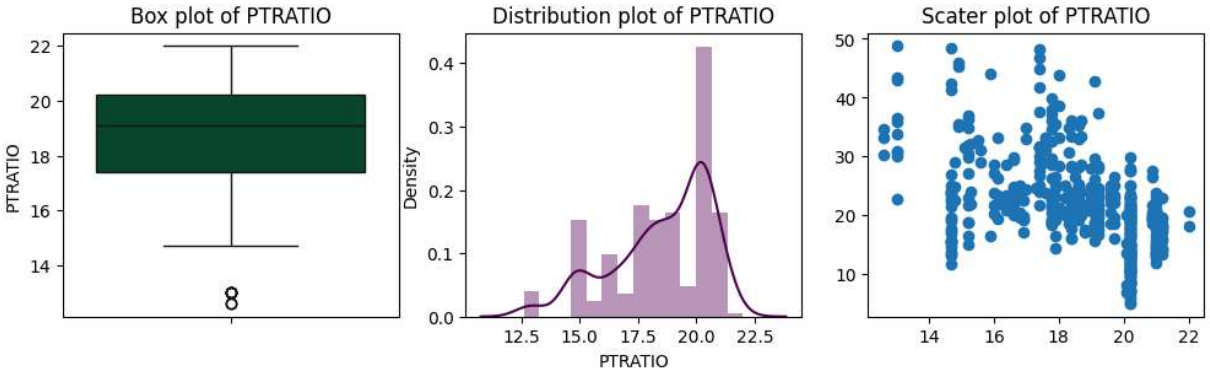
Out[203…

	RM	TAX	PTRATIO	LSTAT	MEDV
54	5.888	469.0	21.1	14.80	18.9
111	6.715	432.0	17.8	10.16	22.8
112	5.913	432.0	17.8	16.21	18.8
113	6.092	432.0	17.8	17.09	18.7
114	6.254	432.0	17.8	10.45	18.5
115	5.928	432.0	17.8	15.76	18.3
116	6.176	432.0	17.8	11.43	21.2
117	6.021	432.0	17.8	10.30	19.2
118	5.872	432.0	17.8	15.37	20.4
119	5.731	432.0	17.8	13.61	19.3
127	5.693	437.0	21.2	17.19	16.2
128	6.431	437.0	21.2	15.39	18.0
129	5.637	437.0	21.2	18.34	14.3
130	6.458	437.0	21.2	12.60	19.2
131	6.326	437.0	21.2	12.26	19.6
132	6.372	437.0	21.2	11.12	23.0
133	5.822	437.0	21.2	15.03	18.4
134	5.757	437.0	21.2	17.31	15.6
135	6.335	437.0	21.2	16.96	18.1
136	5.942	437.0	21.2	16.90	17.4
137	6.454	437.0	21.2	14.59	17.1
138	5.857	437.0	21.2	21.32	13.3
139	6.151	437.0	21.2	18.46	17.8
140	6.174	437.0	21.2	24.16	14.0
141	5.019	437.0	21.2	34.41	14.4
328	5.868	430.0	16.9	9.97	19.3
329	6.333	430.0	16.9	7.34	22.6
330	6.144	430.0	16.9	9.09	19.8

In [204…

```
plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
```

```
sns.boxplot(df2.PTRATIO,color='#005030')
plt.title('Box plot of PTRATIO')

plt.subplot(1,3,2)
sns.distplot(a=df2.PTRATIO,color='#500050')
plt.title('Distribution plot of PTRATIO')

plt.subplot(1,3,3)
plt.scatter(y=df2.MEDV, x=df2.PTRATIO)
plt.title('Scater plot of PTRATIO')

plt.show()
```



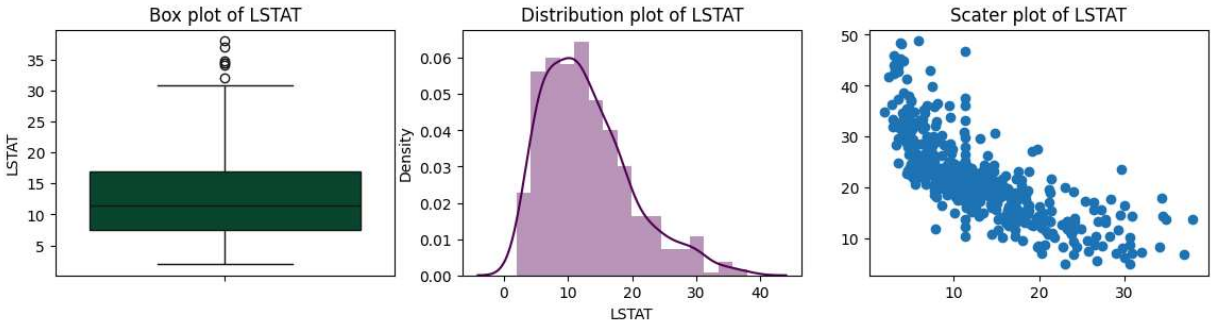
In [205…
```
df2[df2['PTRATIO']<14].sort_values(by=['TAX','MEDV'])
```

Out[205…

	RM	TAX	PTRATIO	LSTAT	MEDV
265	5.560	264.0	13.0	10.45	22.8
259	6.842	264.0	13.0	6.90	30.1
266	7.014	264.0	13.0	14.79	30.7
263	7.327	264.0	13.0	11.25	31.0
260	7.203	264.0	13.0	9.59	33.8
258	7.333	264.0	13.0	7.79	36.0
264	7.206	264.0	13.0	8.10	36.5
261	7.520	264.0	13.0	7.26	43.1
268	7.470	264.0	13.0	3.16	43.5
262	8.398	264.0	13.0	5.91	48.8
197	7.107	329.0	12.6	8.61	30.3
196	7.287	329.0	12.6	4.08	33.3
198	7.274	329.0	12.6	6.62	34.6

In [206…
```
plt.figure(figsize=(14,3))

plt.subplot(1,3,1)
```

```
sns.boxplot(df2.LSTAT,color='#005030')
plt.title('Box plot of LSTAT')

plt.subplot(1,3,2)
sns.distplot(a=df2.LSTAT,color='#500050')
plt.title('Distribution plot of LSTAT')

plt.subplot(1,3,3)
plt.scatter(y=df2.MEDV, x=df2.LSTAT)
plt.title('Scater plot of LSTAT')

plt.show()
```



In [207…

```
D2 = df2.describe().round(2)
D2
```

Out[207…

	RM	TAX	PTRATIO	LSTAT	MEDV
count	489.00	489.00	489.00	489.00	489.00
mean	6.24	317.81	18.52	12.95	21.62
std	0.65	58.08	2.11	6.95	7.86
min	3.56	188.00	12.60	1.98	5.00
25%	5.88	281.00	17.40	7.56	16.70
50%	6.18	330.00	19.10	11.43	20.90
75%	6.58	338.64	20.20	16.94	24.60
max	8.78	469.00	22.00	37.97	48.80

In [208…

```
LSTAT_Q3=D2['LSTAT']['75%']
LSTAT_Q1=D2['LSTAT']['25%']
LSTAT_IQR=LSTAT_Q3-LSTAT_Q1
LSTAT_LV=LSTAT_Q1-1.5*LSTAT_IQR
LSTAT_UV=LSTAT_Q3+1.5*LSTAT_IQR
```

In [209…

```
df2[df2['LSTAT']>LSTAT_UV]
```

Out[209…

	RM	TAX	PTRATIO	LSTAT	MEDV
141	5.019	437.0	21.2	34.41	14.4
373	4.906	335.0	20.2	34.77	13.8
374	4.138	335.0	20.2	37.97	13.8
387	5.000	335.0	20.2	31.99	7.4
412	4.628	335.0	20.2	34.37	17.9
414	4.519	335.0	20.2	36.98	7.0
438	5.935	335.0	20.2	34.02	8.4

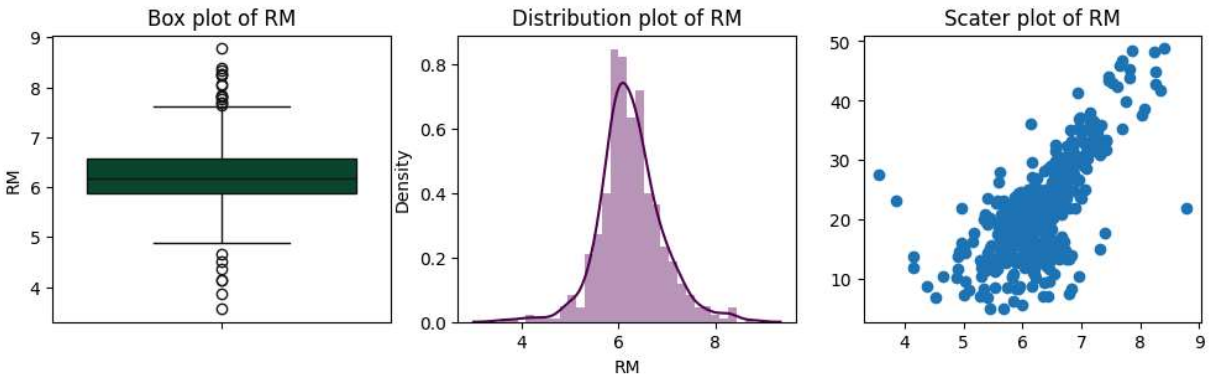
In [210…
```
df2.shape
```

Out[210… (489, 5)

In [211…
```
df2 = df2.drop(index=412, axis=0)
df2.shape
```

Out[211… (488, 5)

In [212…
```
plt.figure(figsize=(12,3))

plt.subplot(1,3,1)
sns.boxplot(df2.RM,color='#005030')
plt.title('Box plot of RM')

plt.subplot(1,3,2)
sns.distplot(a=df2.RM,color='#500050')
plt.title('Distribution plot of RM')

plt.subplot(1,3,3)
plt.scatter(y=df2.MEDV, x=df2.RM)
plt.title('Scater plot of RM')

plt.show()
```



In [213…
```
D3 = df2.describe().round(2)
D3
```

Out[213...

	RM	TAX	PTRATIO	LSTAT	MEDV
count	488.00	488.00	488.00	488.00	488.00
mean	6.25	317.77	18.52	12.91	21.63
std	0.65	58.14	2.11	6.89	7.87
min	3.56	188.00	12.60	1.98	5.00
25%	5.88	280.75	17.40	7.55	16.68
50%	6.18	330.00	19.10	11.43	20.90
75%	6.58	338.64	20.20	16.91	24.62
max	8.78	469.00	22.00	37.97	48.80

In [214...

```
RM_Q1 = D3['RM']['25%']
RM_Q3 = D3['RM']['75%']
RM_IQR = RM_Q3 - RM_Q1
RM_UV = RM_Q3 + 1.5*RM_IQR
RM_LV = RM_Q1 - 1.5*RM_IQR
```

In [215...

```
df2[df2['RM']>RM_UV]
```

Out[215...

	RM	TAX	PTRATIO	LSTAT	MEDV
97	8.069	276.000000	18.0	4.21	38.7
98	7.820	276.000000	18.0	3.57	43.8
180	7.765	193.000000	17.8	7.56	39.8
203	7.853	224.000000	14.7	3.81	48.5
224	8.266	307.000000	17.4	4.14	44.8
226	8.040	307.000000	17.4	11.43	37.6
228	7.686	307.000000	17.4	11.43	46.7
232	8.337	307.000000	17.4	2.47	41.7
233	8.247	307.000000	17.4	3.95	48.3
253	8.259	330.000000	19.1	3.54	42.8
262	8.398	264.000000	13.0	5.91	48.8
273	7.691	223.000000	18.6	6.58	35.2
280	7.820	216.000000	14.9	3.76	45.4
282	7.645	216.000000	14.9	3.01	46.0
364	8.780	294.139785	20.2	5.29	21.9

In [216…
```
df2.shape
```

Out[216…
```
(488, 5)
```

In [217…
```
df2 = df2.drop(index=364)
df2.shape
```

Out[217…
```
(487, 5)
```

In [218…
```
df2[df2['RM']<RM_LV]
```

Out[218…

	RM	TAX	PTRATIO	LSTAT	MEDV
365	3.561	294.139785	20.2	7.12	27.5
367	3.863	330.770270	20.2	13.33	23.1
374	4.138	335.000000	20.2	37.97	13.8
384	4.368	335.000000	20.2	30.63	8.8
386	4.652	338.636364	20.2	28.28	10.5
406	4.138	338.636364	20.2	23.34	11.9
414	4.519	335.000000	20.2	36.98	7.0

In [219…
```
df2.shape
```

Out[219…
```
(487, 5)
```

In [220…
```
df2 = df2.drop(index=[365,367])
df2.shape
```

Out[220…
```
(485, 5)
```

In [221…
```
x = df2[['RM','TAX','PTRATIO','LSTAT']]
y = df2[['MEDV']]

print(x.shape)
print(y.shape)
```

```
(485, 4)
(485, 1)
```

In [222…
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)
print(x)
```

```
[[ 0.51931432 -0.37512179 -1.51917801 -1.15477709]
 [ 0.27092115 -1.30240921 -0.33575152 -0.5509799 ]
 [ 1.50320933 -1.30240921 -0.33575152 -1.29266347]
 ...
 [ 1.16610432 -0.77007754  1.1790344  -1.05898235]
 [ 0.87254876 -0.77007754  1.1790344  -0.93706176]
 [-0.35973942 -0.77007754  1.1790344  -0.73386078]]
```

In [223…
```
m,n = x.shape
x = np.append(arr=np.ones((m,1)), values=x, axis=1)
```

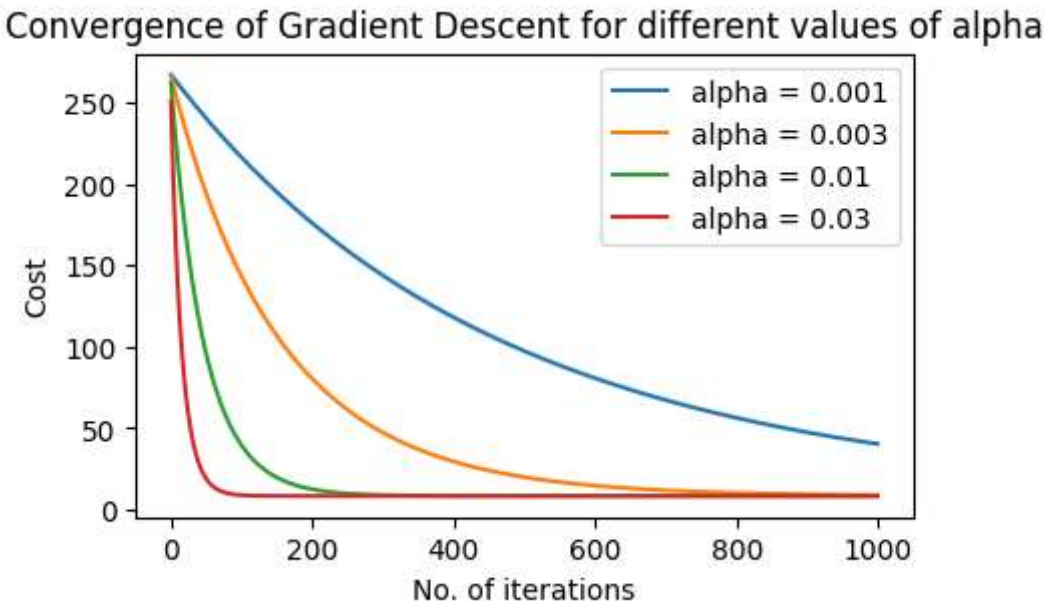
In [224…
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=2)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(363, 5)
(122, 5)
(363, 1)
(122, 1)
```

In [225…
```
def ComputeCost(X,y,theta):
    m=X.shape[0] #number of data points in the set
    J = (1/(2*m)) * np.sum((X.dot(theta) - y)**2)
    return J

#Gradient Descent Algorithm to minimize the Cost and find best parameters in order
def GradientDescent(X,y,theta,alpha,no_of_iters):
    m=X.shape[0]
    J_Cost = []
    for i in range(no_of_iters):
        error = np.dot(X.transpose(),(X.dot(theta)-y))
        theta = theta - alpha * (1/m) * error
        J_Cost.append(ComputeCost(X,y,theta))
    return theta, np.array(J_Cost)
```

In [226…
```
iters = 1000

alpha1 = 0.001
theta1 = np.zeros((x_train.shape[1],1))
theta1, J_Costs1 = GradientDescent(x_train,y_train,theta1,alpha1,iters)

alpha2 = 0.003
theta2 = np.zeros((x_train.shape[1],1))
theta2, J_Costs2 = GradientDescent(x_train,y_train,theta2,alpha2,iters)

alpha3 = 0.01
theta3 = np.zeros((x_train.shape[1],1))
theta3, J_Costs3 = GradientDescent(x_train,y_train,theta3,alpha3,iters)

alpha4 = 0.03
theta4 = np.zeros((x_train.shape[1],1))
theta4, J_Costs4 = GradientDescent(x_train,y_train,theta4,alpha4,iters)
```

In [227…
```
plt.figure(figsize=(5,3))
plt.plot(J_Costs1,label = 'alpha = 0.001')
plt.plot(J_Costs2,label = 'alpha = 0.003')
plt.plot(J_Costs3,label = 'alpha = 0.01')
plt.plot(J_Costs4,label = 'alpha = 0.03')
plt.title('Convergence of Gradient Descent for different values of alpha')
plt.xlabel('No. of iterations')
plt.ylabel('Cost')
plt.legend()
plt.show()
```



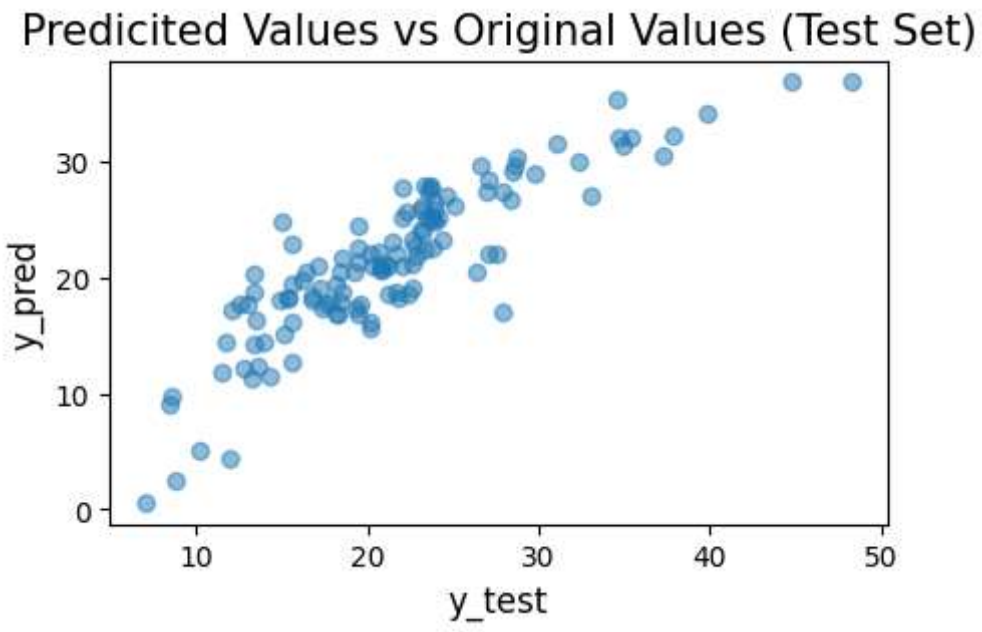
In [228…
```
theta4
```

Out[228…
```
array([[21.63476663],
       [ 3.15692259],
       [-0.89289761],
       [-2.1814731 ],
       [-2.93312318]])
```

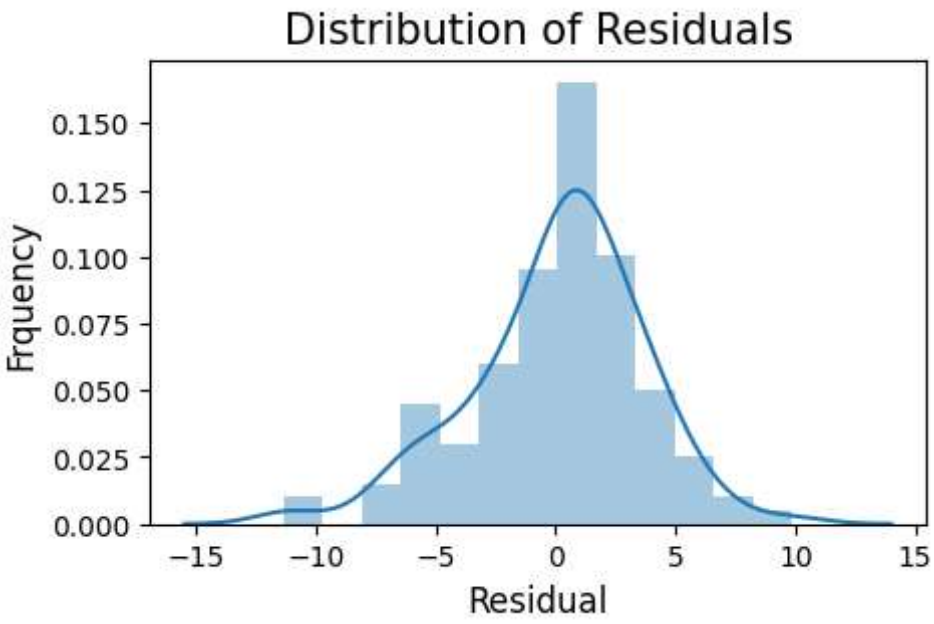
In [229…
```
def Predict(X,theta):
    y_pred = X.dot(theta)
    return y_pred

y_pred = Predict(x_test,theta4)
y_pred[:5]
```

Out[229…
```
array([[20.99397542],
       [26.31692767],
       [36.92529146],
       [24.83468142],
       [14.38903649]])
```

In [230…
```
plt.figure(figsize=(5,3))
plt.scatter(x=y_test,y=y_pred,alpha=0.5)
plt.xlabel('y_test',size=12)
plt.ylabel('y_pred',size=12)
plt.title('Predicited Values vs Original Values (Test Set)',size=15)
plt.show()
```



In [231…
```
plt.figure(figsize=(5,3))
sns.distplot(y_pred-y_test)
plt.xlabel('Residual',size=12)
plt.ylabel('Frquency',size=12)
plt.title('Distribution of Residuals',size=15)
plt.show()
```



In [232…
```
from sklearn import metrics
r2= metrics.r2_score(y_test,y_pred)
N,p = x_test.shape
adj_r2 = 1-((1-r2)*(N-1))/(N-p-1)
print(f'R^2 = {r2}')
print(f'Adjusted R^2 = {adj_r2}')
```

```
R^2 = 0.7563936010569238
Adjusted R^2 = 0.7458933252404119
```

In [233…
```
mse = metrics.mean_squared_error(y_test,y_pred)
mae = metrics.mean_absolute_error(y_test,y_pred)
rmse = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
print(f'Mean Squared Error: {mse}',f'Mean Absolute Error: {mae}',f'Root Mean Square
```

```
Mean Squared Error: 12.790633683786712
Mean Absolute Error: 2.7283932263779804
Root Mean Squared Error: 3.5763995419676915
```

In [234…
```
#coefficients of regression model
coeff=np.array([y for x in theta4 for y in x]).round(2)
features=['Bias','RM','TAX','PTRATIO','LSTAT']
eqn = 'MEDV = '
for f,c in zip(features,coeff):
    eqn+=f" + ({c} * {f})";

print(eqn)
```

```
MEDV =  + (21.63 * Bias) + (3.16 * RM) + (-0.89 * TAX) + (-2.18 * PTRATIO) + (-2.93
* LSTAT)
```

In [235…
```
plt.figure(figsize=(5,3))
sns.barplot(x=features,y=coeff)
plt.ylim([-5,25])
plt.xlabel('Coefficient Names',size=12)
plt.ylabel('Coefficient Values',size=12)
```

```
plt.title('Visualising Regression Coefficients',size=15)
plt.show()
```