In [1]:
```python
# Importing the libraries
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn import linear_model
```

In [2]:
```python
# Importing the Boston Housing dataset
df=pd.read_csv("housing.csv")
df
```

Out[2]:

	0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30 396.90 4.98 24.00
0	0.02731 0.00 7.070 0 0.4690 6.4210 78...
1	0.02729 0.00 7.070 0 0.4690 7.1850 61...
2	0.03237 0.00 2.180 0 0.4580 6.9980 45...
3	0.06905 0.00 2.180 0 0.4580 7.1470 54...
4	0.02985 0.00 2.180 0 0.4580 6.4300 58...
...	...
500	0.06263 0.00 11.930 0 0.5730 6.5930 69...
501	0.04527 0.00 11.930 0 0.5730 6.1200 76...
502	0.06076 0.00 11.930 0 0.5730 6.9760 91...
503	0.10959 0.00 11.930 0 0.5730 6.7940 89...
504	0.04741 0.00 11.930 0 0.5730 6.0300 80...

505 rows × 1 columns

In [3]:
```python
# See head of the dataset
df.head()
```

Out[3]:

	0.00632 18.00 2.310 0 0.5380 6.5750 65.20 4.0900 1 296.0 15.30 396.90 4.98 24.00
0	0.02731 0.00 7.070 0 0.4690 6.4210 78...
1	0.02729 0.00 7.070 0 0.4690 7.1850 61...
2	0.03237 0.00 2.180 0 0.4580 6.9980 45...
3	0.06905 0.00 2.180 0 0.4580 7.1470 54...
4	0.02985 0.00 2.180 0 0.4580 6.4300 58...

In [6]:
```python
#Adding the feature names to the dataframe
df=pd.read_csv("housing12.csv")
df
```

Out[6]:

	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	296.0	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	242.0	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	242.0	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	222.0	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	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273.0	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	273.0	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	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

In [7]:
```python
df.head()
```

Out[7]:

	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	296.0	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	242.0	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	242.0	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	222.0	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	222.0	18.7	396.90	5.33	36.2

In [8]:
```python
df.tail()
```

Out[8]:

	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	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273.0	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	273.0	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	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

In [9]:
```python
df.shape
```

Out[9]:
```
(506, 14)
```

In [10]:
```python
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     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     452 non-null    float64
dtypes: float64(13), int64(1)
memory usage: 55.5 KB
```

In [11]:
```python
df.describe()
```

Out[11]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	452.000000
mean	1.269195	13.295257	9.205158	0.140765	1.101175	15.679800	58.744660	6.173308	78.063241	339.317787	42.614980	332.791107	11.537806	23.750442
std	2.399207	23.048697	7.169630	0.312765	1.646991	27.220206	33.104049	6.476435	203.542157	180.670077	87.585243	125.322456	6.064932	8.808602
min	0.000000	0.000000	0.000000	0.000000	0.385000	3.561000	1.137000	1.129600	1.000000	20.200000	2.600000	0.320000	1.730000	6.300000
25%	0.049443	0.000000	3.440000	0.000000	0.449000	5.961500	32.000000	2.430575	4.000000	254.000000	17.000000	364.995000	6.877500	18.500000
50%	0.144655	0.000000	6.960000	0.000000	0.538000	6.322500	65.250000	3.925850	5.000000	307.000000	18.900000	390.660000	10.380000	21.950000
75%	0.819623	18.100000	18.100000	0.000000	0.647000	6.949000	89.975000	6.332075	24.000000	403.000000	20.200000	395.615000	15.015000	26.600000
max	9.966540	100.000000	27.740000	1.000000	7.313000	100.000000	100.000000	24.000000	666.000000	711.000000	396.900000	396.900000	34.410000	50.000000

In [12]:
```python
df.isnull()
```

Out[12]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	False	False	False	False	False	False	False	False	False	False	False	False	False	False
502	False	False	False	False	False	False	False	False	False	False	False	False	False	False
503	False	False	False	False	False	False	False	False	False	False	False	False	False	False
504	False	False	False	False	False	False	False	False	False	False	False	False	False	False
505	False	False	False	False	False	False	False	False	False	False	False	False	False	False

506 rows × 14 columns

In [14]:
```python
df.dtypes
```

Out[14]:
```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

In [15]:
```python
# Identifying the unique number of values in the dataset
df.nunique()
```

Out[15]:
```
CRIM       452
ZN          27
INDUS       77
CHAS        16
NOX        132
RM         437
AGE        399
DIS        361
RAD         10
TAX         67
PTRATIO     85
B          374
LSTAT      445
MEDV       210
dtype: int64
```

In [16]:
```python
# Check for missing values
df.isnull().sum()
```

Out[16]:
```
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       54
dtype: int64
```

In [17]:
```python
# See rows with missing values
df[df.isnull().any(axis=1)]
```

Out[17]:

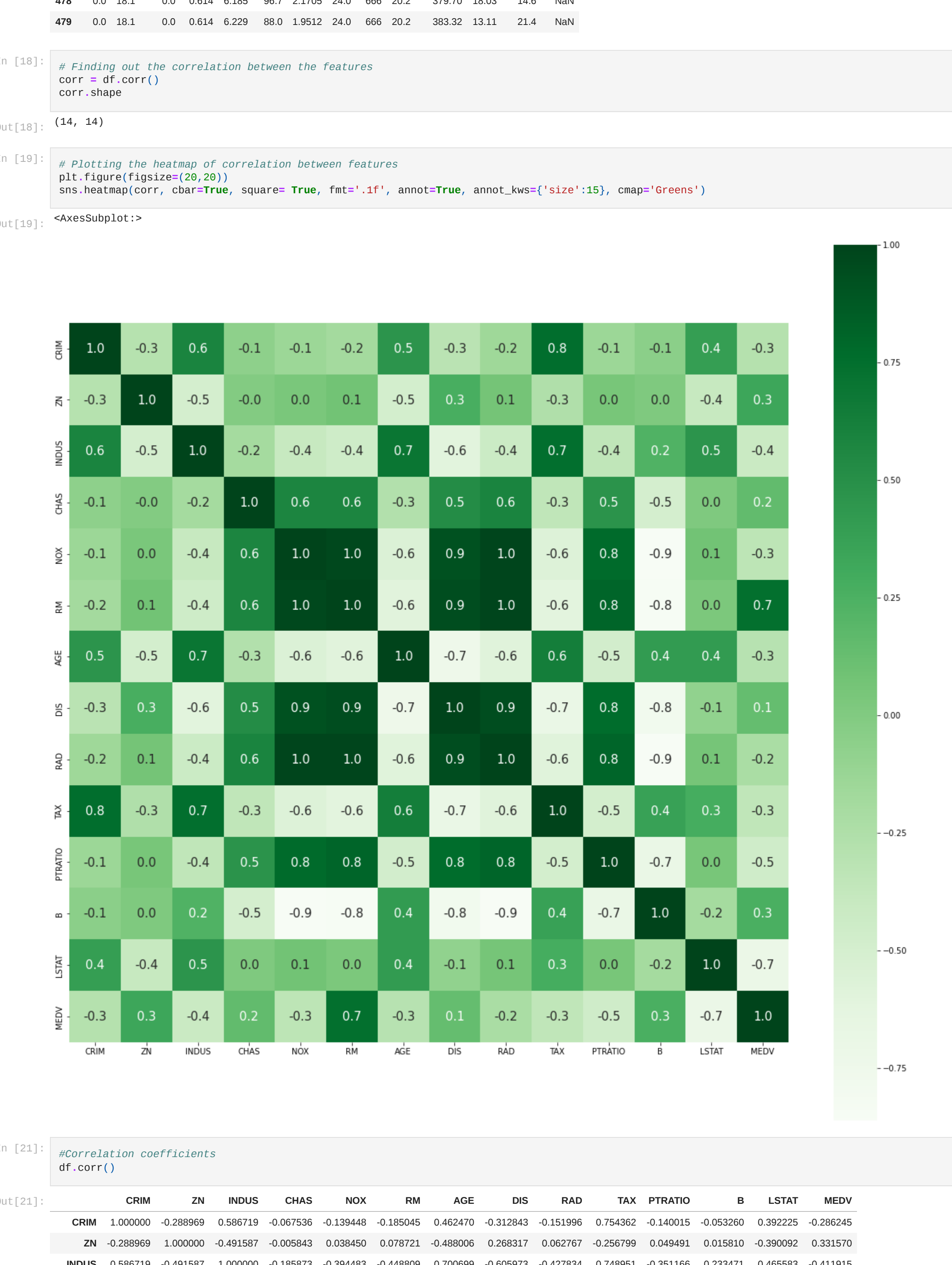
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
367	0.0	18.1	0.0	0.631	3.863	100.0	1.5106	24.0	666	20.2	131.42	13.33	23.1	NaN
373	0.0	18.1	0.0	0.668	4.906	100.0	1.1742	24.0	666	20.2	396.90	34.77	13.8	NaN
374	0.0	18.1	0.0	0.668	4.138	100.0	1.1370	24.0	666	20.2	396.90	37.97	13.8	NaN
375	0.0	18.1	0.0	0.671	7.313	97.9	1.3163	24.0	666	20.2	396.90	13.44	15.0	NaN
376	0.0	18.1	0.0	0.671	6.649	93.3	1.3449	24.0	666	20.2	363.02	23.24	13.9	NaN
378	0.0	18.1	0.0	0.671	6.380	96.2	1.3861	24.0	666	20.2	396.90	23.69	13.1	NaN
379	0.0	18.1	0.0	0.671	6.223	100.0	1.3861	24.0	666	20.2	393.74	21.78	10.2	NaN
380	0.0	18.1	0.0	0.671	6.968	91.9	1.4165	24.0	666	20.2	396.90	17.21	10.4	NaN
381	0.0	18.1	0.0	0.671	6.545	99.1	1.5192	24.0	666	20.2	396.90	21.08	10.9	NaN
384	0.0	18.1	0.0	0.700	4.368	91.2	1.4395	24.0	666	20.2	285.83	30.63	8.8	NaN
385	0.0	18.1	0.0	0.700	5.277	98.1	1.4261	24.0	666	20.2	396.90	30.81	7.2	NaN
386	0.0	18.1	0.0	0.700	4.652	100.0	1.4672	24.0	666	20.2	396.90	28.28	10.5	NaN
387	0.0	18.1	0.0	0.700	5.000	89.5	1.5184	24.0	666	20.2	396.90	31.99	7.4	NaN
388	0.0	18.1	0.0	0.700	4.880	100.0	1.5895	24.0	666	20.2	372.92	30.62	10.2	NaN
392	0.0	18.1	0.0	0.700	5.036	97.0	1.7700	24.0	666	20.2	396.90	25.68	9.7	NaN
394	0.0	18.1	0.0	0.693	5.887	94.7	1.7821	24.0	666	20.2	396.90	16.35	12.7	NaN
398	0.0	18.1	0.0	0.693	5.453	100.0	1.4896	24.0	666	20.2	396.90	30.59	5.0	NaN
400	0.0	18.1	0.0	0.693	5.987	100.0	1.5888	24.0	666	20.2	396.90	26.77	5.6	NaN
401	0.0	18.1	0.0	0.693	6.343	100.0	1.5741	24.0	666	20.2	396.90	20.32	7.2	NaN
403	0.0	18.1	0.0	0.693	5.349	96.0	1.7028	24.0	666	20.2	396.90	19.77	8.3	NaN
404	0.0	18.1	0.0	0.693	5.531	85.4	1.6074	24.0	666	20.2	329.46	27.38	8.5	NaN
405	0.0	18.1	0.0	0.693	5.683	100.0	1.4254	24.0	666	20.2	384.97	22.98	5.0	NaN
406	0.0	18.1	0.0	0.659	4.138	100.0	1.1781	24.0	666	20.2	370.22	23.34	11.9	NaN
407	0.0	18.1	0.0	0.659	5.608	100.0	1.2852	24.0	666	20.2	332.09	12.13	27.9	NaN
409	0.0	18.1	0.0	0.597	6.852	100.0	1.4655	24.0	666	20.2	179.36	19.78	27.5	NaN
410	0.0	18.1	0.0	0.597	5.757	100.0	1.4130	24.0	666	20.2	2.60	10.11	15.0	NaN
411	0.0	18.1	0.0	0.597	6.657	100.0	1.5275	24.0	666	20.2	35.05	21.22	17.2	NaN
412	0.0	18.1	0.0	0.597	4.628	100.0	1.5539	24.0	666	20.2	28.79	34.37	17.9	NaN
413	0.0	18.1	0.0	0.597	5.155	100.0	1.5894	24.0	666	20.2	210.97	20.08	16.3	NaN
414	0.0	18.1	0.0	0.693	4.519	100.0	1.6582	24.0	666	20.2	88.27	36.98	7.0	NaN
415	0.0	18.1	0.0	0.679	6.434	100.0	1.8347	24.0	666	20.2	27.25	29.05	7.2	NaN
416	0.0	18.1	0.0	0.679	6.782	90.8	1.8195	24.0	666	20.2	21.57	25.79	7.5	NaN
417	0.0	18.1	0.0	0.679	5.304	89.1	1.6475	24.0	666	20.2	127.36	26.64	10.4	NaN
418	0.0	18.1	0.0	0.679	5.957	100.0	1.8026	24.0	666	20.2	16.45	20.62	8.8	NaN
419	0.0	18.1	0.0	0.718	6.824	76.5	1.7940	24.0	666	20.2	48.45	22.74	8.4	NaN
420	0.0	18.1	0.0	0.718	6.411	100.0	1.8589	24.0	666	20.2	318.75	15.02	16.7	NaN
422	0.0	18.1	0.0	0.614	5.648	87.6	1.9512	24.0	666	20.2	291.55	14.10	20.8	NaN
425	0.0	18.1	0.0	0.679	5.896	95.4	1.9096	24.0	666	20.2	7.68	24.39	8.3	NaN
426	0.0	18.1	0.0	0.584	5.837	59.7	1.9976	24.0	666	20.2	24.65	15.69	10.2	NaN
427	0.0	18.1	0.0	0.679	6.202	78.7	1.8629	24.0	666	20.2	18.82	14.52	10.9	NaN
431	0.0	18.1	0.0	0.584	6.833	94.3	2.0882	24.0	666	20.2	81.33	19.69	14.1	NaN
434	0.0	18.1	0.0	0.713	6.208	95.0	2.2222	24.0	666	20.2	100.63	15.17	11.7	NaN
435	0.0	18.1	0.0	0.740	6.629	94.6	2.1247	24.0	666	20.2	109.85	23.27	13.4	NaN
436	0.0	18.1	0.0	0.740	6.461	93.3	2.0026	24.0	666	20.2	27.49	18.05	9.6	NaN
437	0.0	18.1	0.0	0.740	6.152	100.0	1.9142	24.0	666	20.2	9.32	26.45	8.7	NaN
438	0.0	18.1	0.0	0.740	5.935	87.9	1.8206	24.0	666	20.2	68.95	34.02	8.4	NaN
440	0.0	18.1	0.0	0.740	5.818	92.4	1.8662	24.0	666	20.2	391.45	22.11	10.5	NaN
444	0.0	18.1	0.0	0.740	5.854	96.6	1.8956	24.0	666	20.2	240.52	23.79	10.8	NaN
445	0.0	18.1	0.0	0.740	6.459	94.8	1.9879	24.0	666	20.2	43.06	23.98	11.8	NaN
468	0.0	18.1	0.0	0.580	5.926	71.0	2.9084	24.0	666	20.2	368.74	18.13	19.1	NaN
469	0.0	18.1	0.0	0.580	5.713	56.7	2.8237	24.0	666	20.2	396.90	14.76	20.1	NaN
477	0.0	18.1	0.0	0.614	5.304	97.3	2.1007	24.0	666	20.2	349.48	24.91	12.0	NaN
478	0.0	18.1	0.0	0.614	6.185	96.7	2.1705	24.0	666	20.2	379.70	18.03	14.6	NaN
479	0.0	18.1	0.0	0.614	6.229	88.0	1.9512	24.0	666	20.2	383.32	13.11	21.4	NaN

In [18]:
```python
# Finding out the correlation between the features
corr = df.corr()
corr.shape
```

Out[18]:
```
(14, 14)
```

In [19]:
```python
# Plotting the heatmap of correlation between features
plt.figure(figsize=(20,20))
sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws={'size':15}, cmap='Greens')
```

Out[19]:
```
<AxesSubplot:>
```



In [21]:
```python
#Correlation coefficients
df.corr()
```

Out[21]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1.000000	-0.288969	0.586719	-0.067536	-0.139448	-0.185045	0.462470	-0.312843	-0.151996	0.754362	-0.140015	-0.053260	0.392225	-0.286245
ZN	-0.288969	1.000000	-0.491587	-0.005843	0.038450	0.078721	-0.488006	0.268317	0.062767	-0.256799	0.049491	0.015810	-0.390092	0.331570
INDUS	0.586719	-0.491587	1.000000	-0.185873	-0.394483	-0.448809	0.700699	-0.605973	-0.427834	0.748951	-0.351166	0.233471	0.465583	-0.411915
CHAS	-0.067536	-0.005843	-0.185873	1.000000	0.585243	0.585447	-0.269616	0.523772	0.587673	-0.347552	0.470870	-0.495956	0.011260	0.154409
NOX	-0.139448	0.038450	-0.394483	0.585243	1.000000	0.975767	-0.550065	0.923503	0.985957	-0.570346	0.775302	-0.856608	0.079688	-0.332778
RM	-0.185045	0.078721	-0.448809	0.585447	0.975767	1.000000	-0.595786	0.946946	0.992620	-0.610962	0.806522	-0.848289	0.029450	0.740181
AGE	0.462470	-0.488006	0.700699	-0.269616	-0.550065	-0.595786	1.000000	-0.744068	-0.585574	0.635697	-0.477048	0.417216	0.414354	-0.299893
DIS	-0.312843	0.268317	-0.605973	0.523772	0.923503	0.946946	-0.744068	1.000000	0.947606	-0.689224	0.766224	-0.778075	-0.080368	0.138798
RAD	-0.151996	0.062767	-0.427834	0.587673	0.985957	0.992620	-0.585574	0.947606	1.000000	-0.586540	0.805556	-0.861694	0.056185	-0.217902
TAX	0.754362	-0.256799	0.748951	-0.347552	-0.570346	-0.610962	0.635697	-0.689224	-0.586540	1.000000	-0.485166	0.372806	0.284030	-0.345898
PTRATIO	-0.140015	0.049491	-0.351166	0.470870	0.775302	0.806522	-0.477048	0.766224	0.805556	-0.485166	1.000000	-0.690245	0.049208	-0.461214
B	-0.053260	0.015810	0.233471	-0.495956	-0.856608	-0.848289	0.417216	-0.778075	-0.861694	0.372806	-0.690245	1.000000	-0.186021	0.264797
LSTAT	0.392225	-0.390092	0.465583	0.011260	0.079688	0.029450	0.414354	-0.080368	0.056185	0.284030	0.049208	-0.186021	1.000000	-0.706255
MEDV	-0.286245	0.331570	-0.411915	0.154409	-0.332778	0.740181	-0.299893	0.138798	-0.217902	-0.345898	-0.461214	0.264797	-0.706255	1.000000

In [22]:
```python
from sklearn.linear_model import LinearRegression
```

In [29]:
```python
df1 = pd.read_csv("https://raw.githubusercontent.com/akshaykhadse/ml-linear-regression/master/data/test.csv")
```

In [30]:
```python
df1
```

Out[30]:

	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0	0.10612	30.0	4.93	0	0.428	6.095	65.1	6.3361	6	300.0	16.6	394.62	12.40
1	1	0.34109	0.0	7.38	0	0.493	6.415	40.1	4.7211	5	287.0	19.6	396.90	6.12
2	2	12.24720	0.0	18.10	0	0.584	5.837	59.7	1.9976	24	666.0	20.2	24.65	15.69
3	3	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311.0	15.2	392.52	20.45
4	4	1.80028	0.0	19.58	0	0.605	5.877	79.2	2.4259	5	403.0	14.7	227.61	12.14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
100	100	0.19073	22.0	5.86	0	0.431	6.718	17.5	7.8265	7	330.0	19.1	393.74	6.56
101	101	6.96215	0.0	18.10	0	0.700	5.713	97.0	1.9265	24	666.0	20.2	394.43	17.11
102	102	0.05360	21.0	5.64	0	0.439	6.511	21.1	6.8147	4	243.0	16.8	396.90	5.28
103	103	0.10469	40.0	6.41	1	0.447	7.267	49.0	4.7872	4	254.0	17.6	389.25	6.05
104	104	4.55587	0.0	18.10	0	0.718	3.561	87.9	1.6132	24	666.0	20.2	354.70	7.12

105 rows × 14 columns

In [31]:
```python
df1.head()
```

Out[31]:

	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0	0.10612	30.0	4.93	0	0.428	6.095	65.1	6.3361	6	300.0	16.6	394.62	12.40
1	1	0.34109	0.0	7.38	0	0.493	6.415	40.1	4.7211	5	287.0	19.6	396.90	6.12
2	2	12.24720	0.0	18.10	0	0.584	5.837	59.7	1.9976	24	666.0	20.2	24.65	15.69
3	3	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311.0	15.2	392.52	20.45
4	4	1.80028	0.0	19.58	0	0.605	5.877	79.2	2.4259	5	403.0	14.7	227.61	12.14

In [32]:
```python
df2 = pd.read_csv("https://raw.githubusercontent.com/akshaykhadse/ml-linear-regression/master/data/train.csv")
```

In [33]:
```python
df2
```

Out[33]:

	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0	0.95577	0.0	8.14	0	0.538	6.047	88.8	4.4534	4	307.0	21.0	306.38	17.28	14.8
1	1	0.02875	28.0	15.04	0	0.464	6.211	28.9	3.6659	4	270.0	18.2	396.33	6.21	25.0
2	2	1.22358	0.0	19.58	0	0.605	6.943	97.4	1.8773	5	403.0	14.7	363.43	4.59	41.3
3	3	5.66637	0.0	18.10	0	0.740	6.219	100.0	2.0048	24	666.0	20.2	395.69	16.59	18.4
4	4	0.04544	0.0	3.24	0	0.460	6.144	32.2	5.8736	4	430.0	16.9	368.57	9.09	19.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	395	0.03615	80.0	4.95	0	0.411	6.630	23.4	5.1167	4	245.0	19.2	396.90	4.70	27.9
396	396	0.17505	0.0	5.96	0	0.499	5.966	30.2	3.8473	5	279.0	19.2	393.43	10.13	24.7
397	397	6.65492	0.0	18.10	0	0.713	6.317	83.0	2.7344	24	666.0	20.2	396.90	13.99	19.5
398	398	0.13117	0.0	8.56	0	0.520	6.127	85.2	2.1224	5	384.0	20.9	387.69	14.09	20.4
399	399	0.06466	70.0	2.24	0	0.400	6.345	20.1	7.8278	5	358.0	14.8	368.24	4.97	22.5

400 rows × 15 columns

In [34]:
```python
df2.head()
```

Out[34]:

	ID	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0	0.95577	0.0	8.14	0	0.538	6.047	88.8	4.4534	4	307.0	21.0	306.38	17.28	14.8
1	1	0.02875	28.0	15.04	0	0.464	6.211	28.9	3.6659	4	270.0	18.2	396.33	6.21	25.0
2	2	1.22358	0.0	19.58	0	0.605	6.943	97.4	1.8773	5	403.0	14.7	363.43	4.59	41.3
3	3	5.66637	0.0	18.10	0	0.740	6.219	100.0	2.0048	24	666.0	20.2	395.69	16.59	18.4
4	4	0.04544	0.0	3.24	0	0.460	6.144	32.2	5.8736	4	430.0	16.9	368.57	9.09	19.8

In [57]:
```python
# Import phi from train data set
phi = np.loadtxt('https://raw.githubusercontent.com/akshaykhadse/ml-linear-regression/master/data/train.csv', dtype='float', delimiter=',', skiprow
                 usecols=tuple(range(1, 14)))
```

In [37]:
```python
# Import y from train data set
y = np.loadtxt('https://raw.githubusercontent.com/akshaykhadse/ml-linear-regression/master/data/train.csv', dtype='float', delimiter=',', skiprows=
               usecols=14, ndmin=2)
```

In [38]:
```python
# Import phi_test from test data set
phi_test = np.loadtxt('https://raw.githubusercontent.com/akshaykhadse/ml-linear-regression/master/data/test.csv', dtype='float', delimiter=',',
                      skiprows=1, usecols=tuple(range(1, 14)))
```

In [58]:
```python
# Add a cloloumn of 1s to right of phi and phi_test
phi_test = np.concatenate((phi_test, np.ones((105, 1))), axis=1)
phi = np.concatenate((phi, np.ones((400, 1))), axis=1)
```

In [65]:
```python
# Min Max scaling for phi and phi_test (Feature Engineering)
for i in range(0, 13):
    col_max = max(phi[:, i])
    col_min = min(phi[:, i])
    phi[:, i] = (phi[:, i] - col_min) / (col_max - col_min)
    phi_test[:, i] = (phi_test[:, i] - col_min) / (col_max - col_min)
```

In [43]:
```python
# Log scaling on y
y = np.log(y)
```

In [44]:
```python
# Function to calculate change in error function
def delta_w(p, phi, w):
    if p == 2:
        deltaw = (2 * (np.dot(np.dot(np.transpose(phi), phi), w) -
                       np.dot(np.transpose(phi), y)) +
                  lambd * p * np.power(np.absolute(w), (p - 1)))
    if p < 2 and p > 1:
        deltaw = (2 * (np.dot(np.dot(np.transpose(phi), phi), w) -
                       np.dot(np.transpose(phi), y)) +
                  lambd * p * np.power(np.absolute(w), (p - 1)) * np.sign(w))
    return deltaw
```

In [45]:
```python
# Dictionary containing filenames as keys and p as values
filenames = {'output.csv': 2.0,
             'output_p1.csv': 1.75,
             'output_p2.csv': 1.5,
             'output_p3.csv': 1.3
             }
```

In [48]:
```python
# For each item in this dictionary
for (fname, p) in filenames.items():
    # Set initial w to zeros
    w = np.zeros((14, 1))
```

In [66]:
```python
# For each item in this dictionary
for (fname, p) in filenames.items():
    # Set initial w to zeros
    w = np.zeros((14, 1))

    # Hyperparameter lambda value
    lambd = 0.2

    # Maximum step size
    t = 0.00012

    # Calculate new value of w
    w_new = w - t * delta_w(p, phi, w)

    i = 0
    # Repeat steps until error between consecutive w is less than threshold
    while(np.linalg.norm(w_new-w) > 10 ** -10):
        w = w_new
        w_new = w - t * delta_w(p, phi, w)
        i = i + 1

    # Load values of id
    id_test = np.loadtxt('https://raw.githubusercontent.com/akshaykhadse/ml-linear-regression/master/data/test.csv', dtype='int', delimiter=',',
                         skiprows=1, usecols=0, ndmin=2)
```

In [72]:
```python
reg = linear_model.LinearRegression()
reg.fit(df1[['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','B']],df1.LSTAT)
```

Out[72]:
```
LinearRegression()
```

In [73]:
```python
reg.coef_
```

Out[73]:
```
array([ 0.417071  , -0.01975573, -0.01743688, -0.09044361,  0.19915205,
       -1.99662069,  0.12803679,  0.79823561, -0.39557847,  0.01229235,
        0.50622051, -0.01484531])
```

In [74]:
```python
reg.intercept_
```

Out[74]:
```
6.721193193645522
```