Importing the libraries

In [46]:
```
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing the Boston Housing dataset

In [47]:
```
df=pd.read_csv('Boston.csv')
```

In [21]:
```
boston=df
```

In [25]:
```
data = pd.DataFrame(boston)
```

Importing the Boston Housing dataset.head()

In [ ]:
```
data.head()
```

Check the shape of dataframe

In [ ]:
```
data.shape
```

In [42]:
```
data.columns
```

Out[42]:
```
Index(['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis',
       'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

In [43]:
```
data.dtypes
```

Out[43]:
```
Unnamed: 0      int64
crim          float64
zn            float64
indus         float64
chas            int64
nox           float64
rm            float64
age           float64
dis           float64
rad             int64
tax             int64
ptratio       float64
black         float64
lstat         float64
medv          float64
dtype: object
```

Identifying the unique number of values in the dataset

In [45]:
```
data.nunique()
```

Out[45]:
```
Unnamed: 0    506
crim          504
zn             26
indus          76
chas            2
nox            81
rm            446
age           356
dis           412
rad             9
tax            66
ptratio        46
black         357
lstat         455
medv          229
dtype: int64
```

Check for missing values

In [ ]:
```
data.isnull().sum()
```

See rows with missing values

In [48]:
```
data[data.isnull().any(axis=1)]
```

Out[48]:

Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv

Viewing the data statistics

In [49]:
```
data.describe()
```

Out[49]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	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	506.000000	506.000000
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	1.000000	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%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	379.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	506.000000	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

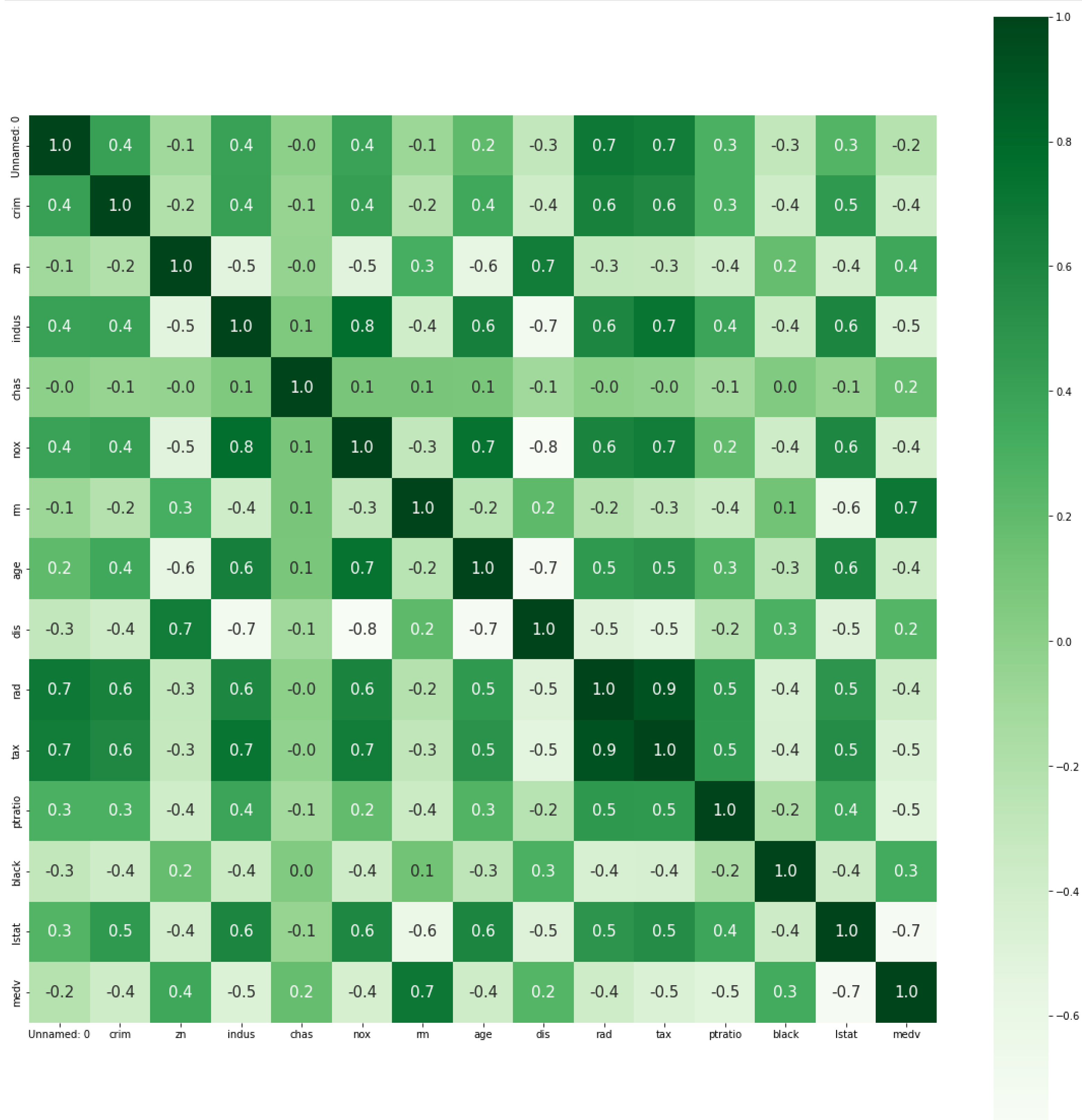
Finding out the correlation between the features

In [50]:
```
corr = data.corr()
corr.shape
```

Out[50]:
```
(15, 15)
```

Plotting the heatmap of correlation between features

In [52]:
```
plt.figure(figsize=(20,20))
sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws={'size':15}, cmap='Greens')
plt.show()
```



Spliting target variable and independent variables

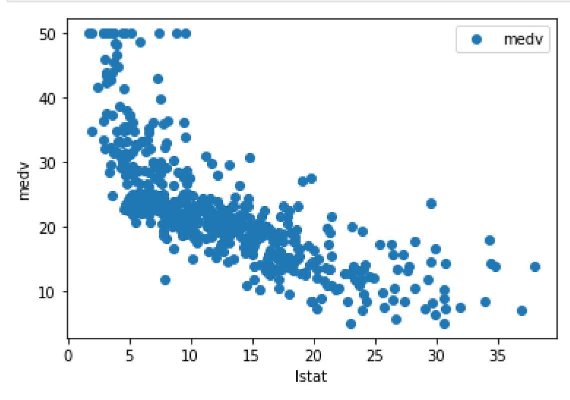
In [54]:
```
data = data.loc[:,['lstat','medv']]  # storing both depep. and indep. into data
data.head()
```

Out[54]:

	lstat	medv
0	4.98	24.0
1	9.14	21.6
2	4.03	34.7
3	2.94	33.4
4	5.33	36.2

Visualizing variables

In [55]:
```
data.plot(x='lstat',y='medv',style='o')
plt.xlabel('lstat')
plt.ylabel('medv')
plt.show()
```



Preparing data

In [59]:
```
X= pd.DataFrame(data['lstat'])
y = pd.DataFrame(data['medv'])
```

In [60]:
```
X.size,  y.size
```

Out[60]:
```
(506, 506)
```

In [92]:
```
print(X_train.size)
print(X_test.size)
print(y_train.size)
print(y_test.size)
```

```
4956
2128
354
152
```

Splitting to training and testing data

In [66]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 4)
```

In [68]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train,y_train)
```

Out[68]:
```
LinearRegression()
```

Value of y intercept

In [69]:
```
print(reg.intercept_)
```

```
[33.64814631]
```

Value of slope

In [70]:
```
print(reg.coef_)
```

```
[[-0.90248701]]
```

Model Evaluation

In [72]:
```
y_pred = reg.predict(X_test)
y_pred = pd.DataFrame(y_pred, columns=['Predicted'])
```

In [73]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error: 4.8399529402570804
Mean Squared Error: 48.69382381816015
Root Mean Squared Error: 6.97809600236054
```

Multiple Linear Regression

In [82]:
```
dataset = pd.read_csv('Boston.csv')
dataset.head()
```

Out[82]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [86]:
```
X  = pd.DataFrame(dataset.iloc[:,:-1])
y = pd.DataFrame(dataset.iloc[:,-1])
```

In [87]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [88]:
```
X_train
```

Out[88]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
5	6	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21
116	117	0.13158	0.0	10.01	0	0.547	6.176	72.5	2.7301	6	432	17.8	393.30	12.04
45	46	0.17142	0.0	6.91	0	0.448	5.682	33.8	5.1004	3	233	17.9	396.90	10.21
16	17	1.05393	0.0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21.0	386.85	6.58
468	469	15.57570	0.0	18.10	0	0.580	5.926	71.0	2.9084	24	666	20.2	368.74	18.13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
106	107	0.17120	0.0	8.56	0	0.520	5.836	91.9	2.2110	5	384	20.9	395.67	18.66
270	271	0.29916	20.0	6.96	0	0.464	5.856	42.1	4.4290	3	223	18.6	388.65	13.00
348	349	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	390.94	5.99
435	436	11.16040	0.0	18.10	0	0.740	6.629	94.6	2.1247	24	666	20.2	109.85	23.27
102	103	0.22876	0.0	8.56	0	0.520	6.405	85.4	2.7147	5	384	20.9	70.80	10.63

354 rows × 14 columns

In [89]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train,y_train)
```

Out[89]:
```
LinearRegression()
```

In [90]:
```
y_pred = reg.predict(X_test)
y_pred = pd.DataFrame(y_pred, columns=['Predicted'])
```

In [91]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error: 3.211664337751415
Mean Squared Error: 21.969368008053873
Root Mean Squared Error: 4.687149241069018
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: