In [1]:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [3]:

```
import matplotlib.pyplot as plt
```

In [4]:

```
import seaborn as sns
```

In [5]:

```
%matplotlib inline
```

In [6]:

```
df = pd.read_csv("Housingdata.csv")
```

In [7]:

```
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	1	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	19.0	7.07	1	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	20.0	7.07	1	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	21.0	2.18	1	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	22.0	2.18	1	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	3.94	36.2

In [8]:

```
df.tail()
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
501	0.06263	289.0	11.93	496	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	12.00	22.4
502	0.04527	290.0	11.93	497	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	291.0	11.93	498	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	292.0	11.93	499	0.573	6.794	92.0	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	293.0	11.93	500	0.573	6.030	93.0	2.5050	1	273	21.0	396.90	7.88	11.9

In [9]:

```
print("The shape of the data is: ")
df.shape
```

```
The shape of the data is: 
```

Out[9]:

```
(506, 14)
```

In [10]:

```
df.isnull().sum()
```

Out[10]:

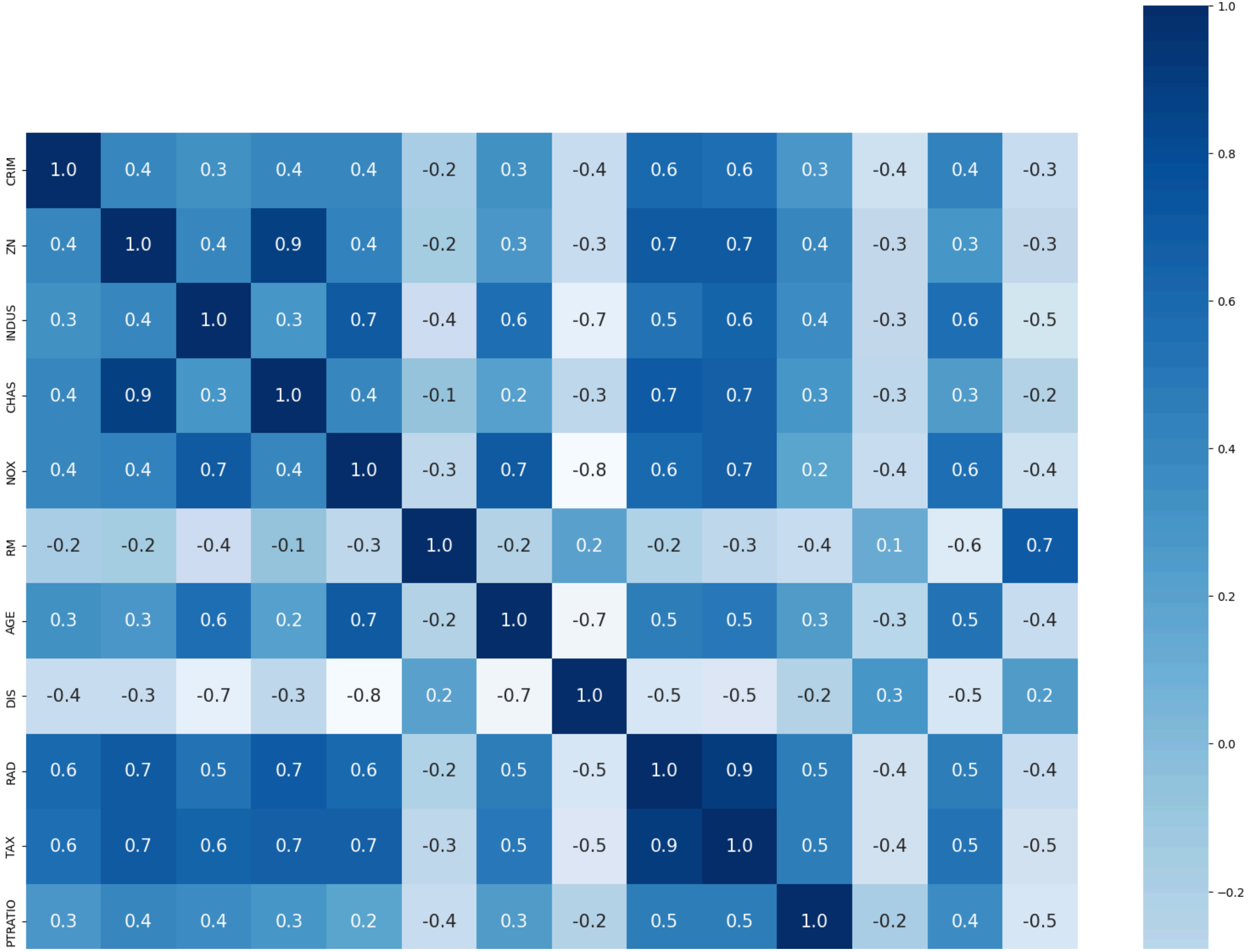
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [11]:

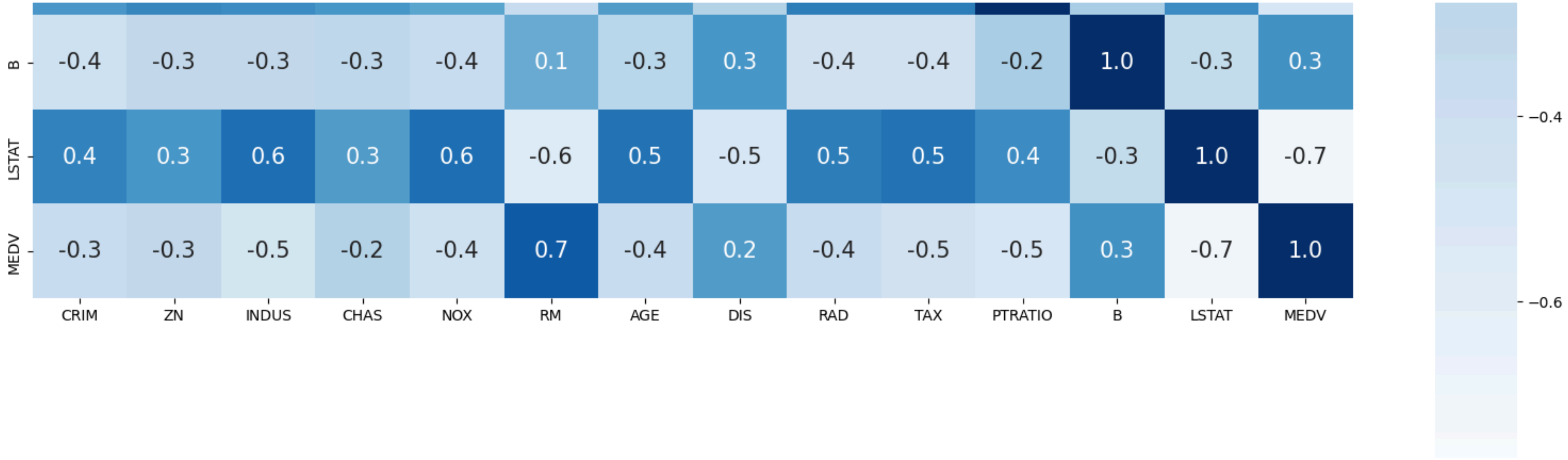
```
corr = df.corr()
corr.shape
```

Out[11]:
```
(14, 14)
```

In [14]:
```
plt.figure(figsize=(20, 20))
sns.heatmap(corr, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size': 15}, cmap='Blues')
plt.show()
```



CRIM	1.0	0.4	0.3	0.4	0.4	-0.2	0.3	-0.4	0.6	0.6	0.3	-0.4	0.4	-0.3
ZN	0.4	1.0	0.4	0.9	0.4	-0.2	0.3	-0.3	0.7	0.7	0.4	-0.3	0.3	-0.3
INDUS	0.3	0.4	1.0	0.3	0.7	-0.4	0.6	-0.7	0.5	0.6	0.4	-0.3	0.6	-0.5
CHAS	0.4	0.9	0.3	1.0	0.4	-0.1	0.2	-0.3	0.7	0.7	0.3	-0.3	0.3	-0.2
NOX	0.4	0.4	0.7	0.4	1.0	-0.3	0.7	-0.8	0.6	0.7	0.2	-0.4	0.6	-0.4
RM	-0.2	-0.2	-0.4	-0.1	-0.3	1.0	-0.2	0.2	-0.2	-0.3	-0.4	0.1	-0.6	0.7
AGE	0.3	0.3	0.6	0.2	0.7	-0.2	1.0	-0.7	0.5	0.5	0.3	-0.3	0.5	-0.4
DIS	-0.4	-0.3	-0.7	-0.3	-0.8	0.2	-0.7	1.0	-0.5	-0.5	-0.2	0.3	-0.5	0.2
RAD	0.6	0.7	0.5	0.7	0.6	-0.2	0.5	-0.5	1.0	0.9	0.5	-0.4	0.5	-0.4
TAX	0.6	0.7	0.6	0.7	0.7	-0.3	0.5	-0.5	0.9	1.0	0.5	-0.4	0.5	-0.5
PTRATIO	0.3	0.4	0.4	0.3	0.2	-0.4	0.3	-0.2	0.5	0.5	1.0	-0.2	0.4	-0.5



In [15]:

```
x = df.drop(['MEDV'], axis = 1)
y = df['MEDV']
```

In [16]:

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size =0.2,random_state = 0)
```

In [17]:

```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

In [18]:

```
model=lm.fit(xtrain,ytrain)
```

In [19]:

```
xtrain
```

Out[19]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
220	0.35809	126.0	6.20	215	0.507	6.951	88.5	2.8617	8	307	17.4	391.70	9.71
71	0.15876	16.5	10.81	66	0.413	5.961	17.5	5.2873	4	305	19.2	376.94	9.88
240	0.11329	30.0	4.93	235	0.428	6.897	54.3	6.3361	6	300	16.6	391.25	11.38
6	0.08829	12.5	7.87	1	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43
417	25.94060	205.0	18.10	412	0.679	5.304	89.1	1.6475	24	666	20.2	127.36	26.64
...	...	...	...	...	...	...	...	...	...	...	...	...	...
323	0.28392	111.0	7.38	318	0.493	5.708	74.3	4.7211	5	287	19.6	391.13	11.74
192	4.12579	98.0	3.44	187	0.437	7.178	26.3	6.4798	5	398	15.2	390.49	2.87
117	0.15098	23.0	10.01	112	0.547	6.021	82.6	2.7474	6	432	17.8	394.51	10.30
47	0.22927	82.0	17.96	42	0.448	6.030	85.5	5.6894	3	233	17.9	392.74	18.80
172	0.13914	78.0	4.05	167	0.510	5.572	88.5	2.5961	5	296	16.6	396.90	14.69

404 rows × 13 columns

In [20]:

```
ytrain_pred=lm.predict(xtrain)
ytest_pred=lm.predict(xtest)
```

In [21]:

```
testdata=[[0.00632,18.0,2.31,0.0,0.538,6.575,65.2,4.0900,1.0,296.0,15.3,396.90,4.98]]
```

In [22]:

```
test_pred = lm.predict(testdata)
test_pred
```

```
/home/pratiksha/.local/lib/python3.8/site-packages/sklearn/base.py:465: UserWarning: X does not have valid feature n
ames, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[22]:
```
array([30.73465291])
```

In [23]:
```
df1=pd.DataFrame(ytrain_pred,ytrain)
df2=pd.DataFrame(ytest_pred,ytest)
df1
```

Out[23]:

	0
MEDV	
26.7	31.083610
21.7	23.425128
22.0	28.744489
22.9	24.368941
10.4	6.805201
...	...
18.5	19.764742
36.4	33.602929
19.2	24.152525
16.6	19.542249
23.1	23.475406

404 rows × 1 columns

In [24]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print('MSE on test data:',mse)
```

```python
mse1 = mean_squared_error(ytrain_pred, ytrain)
print('MSE on training data:',mse1)
```

```
MSE on test data: 36.514221822009354
MSE on training data: 20.6807311647163
```

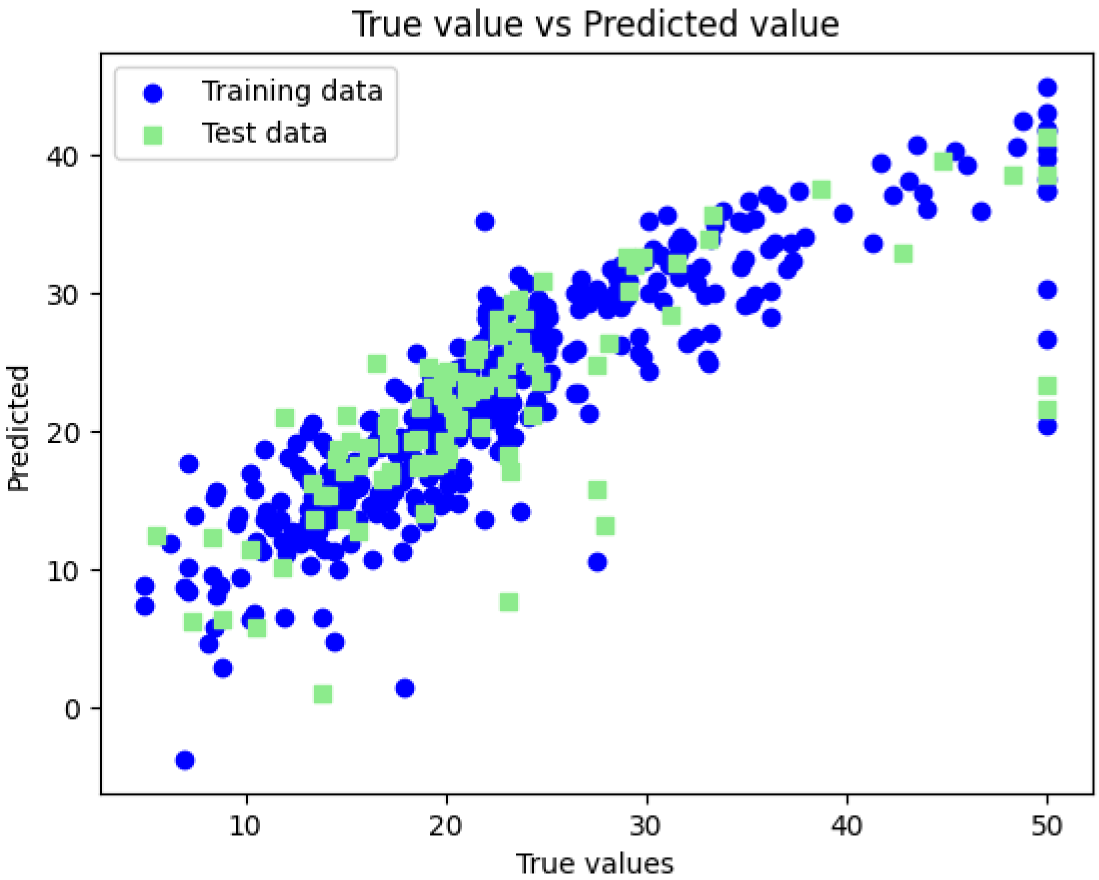
In [25]:

```python
r2 = lm.score(xtest, ytest)
rmse = (np.sqrt(mean_squared_error(ytest, ytest_pred)))
print('r-squared: {}' .format(r2))
print('-----------------------------------------')
print('root mean squared error: {}'.format(rmse))
```

```
r-squared: 0.5515790030771384
-----------------------------------------
root mean squared error: 6.042699878531893
```

In [26]:

```python
plt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data')
plt.scatter(ytest,ytest_pred ,c='lightgreen',marker='s',label='Test data')
plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")
plt.legend(loc= 'upper left')
plt.plot()
plt.show()
```


True value vs Predicted value
● Training data
■ Test data
Predicted
40
30
20
10
0
True values
10
20
30
40
50

In [ ]: