Practical 4

Title: Data Analytics I

```
[ ]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     %matplotlib inline
     import warnings
     warnings.filterwarnings("ignore")
```

```
[ ]: column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',␣
      ↪'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'PRICE']
     data = pd.read_csv('housing.csv', header=None, delimiter=r"\s+",␣
      ↪names=column_names)
     data.head()
```

```
[ ]:       CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
     0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
     1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
     2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
     3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
     4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0

        PTRATIO       B  LSTAT  PRICE
     0     15.3  396.90   4.98   24.0
     1     17.8  396.90   9.14   21.6
     2     17.8  392.83   4.03   34.7
     3     18.7  394.63   2.94   33.4
     4     18.7  396.90   5.33   36.2
```

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

```
[ ]: CRIM       0
     ZN         0
     INDUS      0
     CHAS       0
     NOX        0
     RM         0
     AGE        0
     DIS        0
```

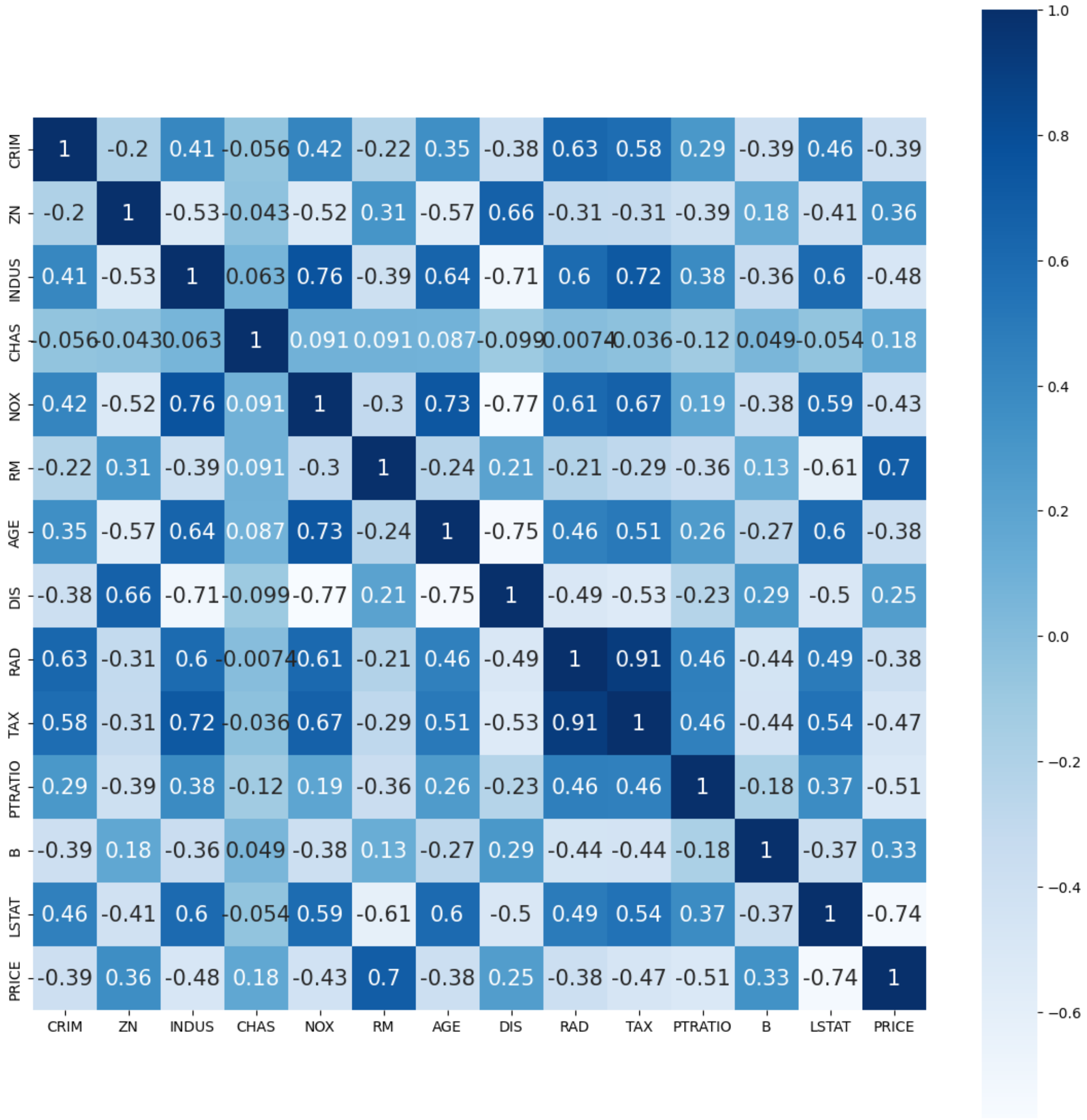
```
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

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

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

```
[ ]: plt.figure(figsize=(14,14))
     sns.heatmap(corr, cbar=True, square= True,  annot=True, annot_kws={'size':15},␣
      ↪cmap='Blues')

     plt.show()
```



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
CRIM	1	-0.2	0.41	-0.056	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39	0.46	-0.39
ZN	-0.2	1	-0.53	-0.043	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
INDUS	0.41	-0.53	1	0.063	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
CHAS	-0.056	-0.043	0.063	1	0.091	0.091	0.087	-0.099	0.0074	0.036	-0.12	0.049	-0.054	0.18
NOX	0.42	-0.52	0.76	0.091	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
RM	-0.22	0.31	-0.39	0.091	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
AGE	0.35	-0.57	0.64	0.087	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
DIS	-0.38	0.66	-0.71	-0.099	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
RAD	0.63	-0.31	0.6	-0.0074	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
TAX	0.58	-0.31	0.72	-0.036	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
PTRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
B	-0.39	0.18	-0.36	0.049	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
LSTAT	0.46	-0.41	0.6	-0.054	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
PRICE	-0.39	0.36	-0.48	0.18	-0.43	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

```
[ ]: x = data.drop(['PRICE'], axis = 1)
     y = data['PRICE']
```

```
[ ]: from sklearn.model_selection import train_test_split
     xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size =0.
      ↪2,random_state = 0)
```

```
[ ]: import sklearn
     from sklearn.linear_model import LinearRegression
     lm = LinearRegression()
     # Train the model using the training sets
     model=lm.fit(xtrain,ytrain)
```

```
[ ]: xtrain
```

```
[ ]:         CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
     220   0.35809   0.0   6.20     1  0.507  6.951  88.5  2.8617    8  307.0
     71    0.15876   0.0  10.81     0  0.413  5.961  17.5  5.2873    4  305.0
     240   0.11329  30.0   4.93     0  0.428  6.897  54.3  6.3361    6  300.0
     6     0.08829  12.5   7.87     0  0.524  6.012  66.6  5.5605    5  311.0
     417  25.94060   0.0  18.10     0  0.679  5.304  89.1  1.6475   24  666.0
     ..        …     …     …    …     …     …     …     …  …     …
     323   0.28392   0.0   7.38     0  0.493  5.708  74.3  4.7211    5  287.0
     192   0.08664  45.0   3.44     0  0.437  7.178  26.3  6.4798    5  398.0
     117   0.15098   0.0  10.01     0  0.547  6.021  82.6  2.7474    6  432.0
     47    0.22927   0.0   6.91     0  0.448  6.030  85.5  5.6894    3  233.0
     172   0.13914   0.0   4.05     0  0.510  5.572  88.5  2.5961    5  296.0

          PTRATIO       B  LSTAT
     220     17.4  391.70   9.71
     71      19.2  376.94   9.88
     240     16.6  391.25  11.38
     6       15.2  395.60  12.43
     417     20.2  127.36  26.64
     ..       …       …     …
     323     19.6  391.13  11.74
     192     15.2  390.49   2.87
     117     17.8  394.51  10.30
     47      17.9  392.74  18.80
     172     16.6  396.90  14.69

     [404 rows x 13 columns]
```

```
[ ]: ytrain_pred=lm.predict(xtrain)
     ytest_pred=lm.predict(xtest)
```

```
[ ]: 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]]
```

```
[ ]: test_pred = lm.predict(testdata)
     test_pred
```

```
[ ]: array([30.49949836])
```

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

```
[ ]:                0
     PRICE
     26.7   32.556927
```

```
21.7   21.927095
22.0   27.543826
22.9   23.603188
10.4    6.571910
…           …
18.5   19.494951
36.4   33.326364
19.2   23.796208
16.6   18.458353
23.1   23.249181

[404 rows x 1 columns]
```

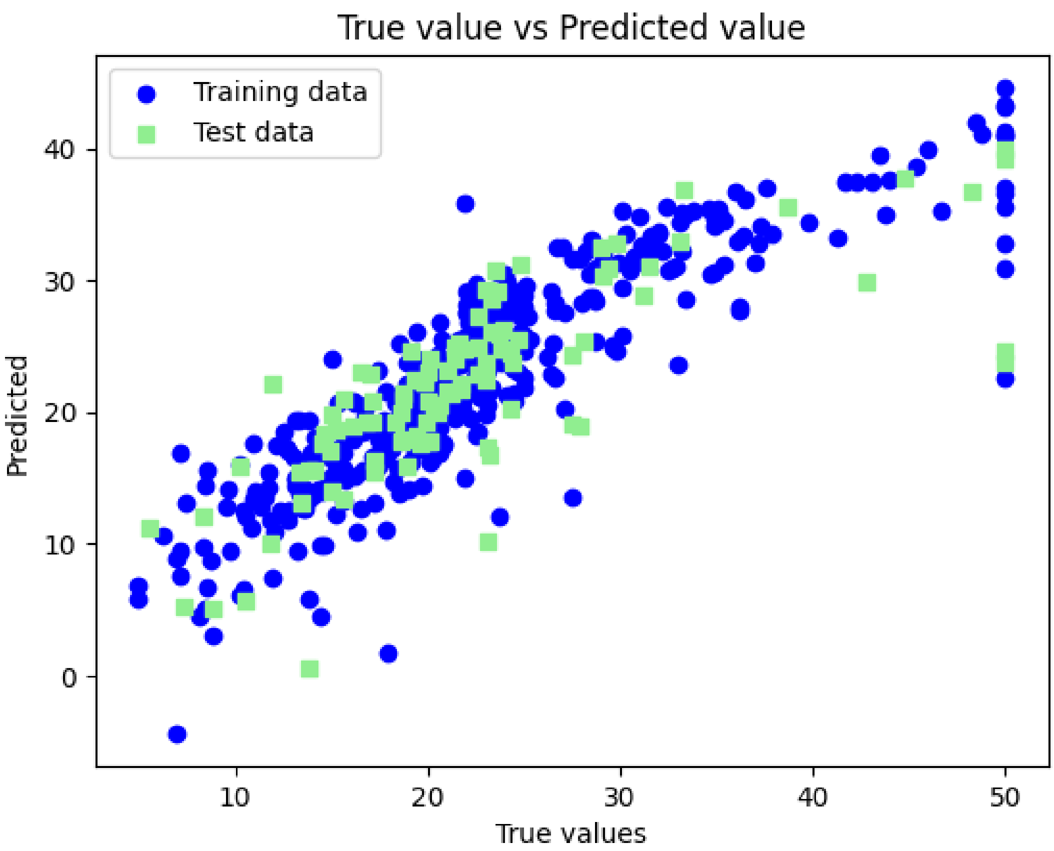
[ ]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print('MSE on test data:',mse)
mse1 = mean_squared_error(ytrain_pred, ytrain)
print('MSE on training data:',mse1)
```

```
MSE on test data: 33.44897999767657
MSE on training data: 19.326470203585725
```

[ ]:
```
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: 0.5892223849182503
-----------------------------------------
root mean squared error: 5.783509315085138
```

[ ]:
```
#plotting the linear regression model
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()
```



```
[ ]: 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]]
```

```
[ ]: test_pred = lm.predict(testdata)
     test_pred
```

```
[ ]: array([30.49949836])
```