

# practical-4

March 10, 2024

```
[ ]: import pandas as pd
import numpy as num
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

```
[ ]: df=pd.read_csv("/content/house1.zip")
```

```
[ ]: df.head()
```

```
[ ]:      crim    zn  indus  chas    nox    rm   age    dis  rad  tax  ptratio  \
0  0.00632  18.0   2.31    0  0.538  6.575  65.2  4.0900    1  296    15.3
1  0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671    2  242    17.8
2  0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671    2  242    17.8
3  0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622    3  222    18.7
4  0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622    3  222    18.7
```

```
      b  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

```
[ ]: df.shape
```

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

```
[ ]: df.info()
df=df.dropna()
df.isnull().sum()
```

```
<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   int64
4   nox     506 non-null   float64
5   rm      501 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   int64
10  ptratio 506 non-null   float64
11  b       506 non-null   float64
12  lstat   506 non-null   float64
13  medv    506 non-null   float64

```

dtypes: float64(11), int64(3)

memory usage: 55.5 KB

```

[ ]: 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

```

```

[ ]: x=df.drop(["medv"],axis=1)
     y = df['medv']

```

```

[ ]: x

```

```

[ ]:
      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
1   0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671   2  242
2   0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671   2  242
3   0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622   3  222
4   0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622   3  222
..      ...   ...   ...   ...   ...   ...   ...   ...   ...
501  0.06263   0.0  11.93    0  0.573  6.593  69.1  2.4786   1  273
502  0.04527   0.0  11.93    0  0.573  6.120  76.7  2.2875   1  273
503  0.06076   0.0  11.93    0  0.573  6.976  91.0  2.1675   1  273

```

```

504  0.10959   0.0  11.93    0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93    0  0.573  6.030  80.8  2.5050    1  273

```

```

      ptratio      b  lstat
0      15.3  396.90   4.98
1      17.8  396.90   9.14
2      17.8  392.83   4.03
3      18.7  394.63   2.94
4      18.7  396.90   5.33
..      ...      ...      ...
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

```

[501 rows x 13 columns]

```
[ ]: y
```

```
[ ]: 0      24.0
      1      21.6
      2      34.7
      3      33.4
      4      36.2

```

```

      ...
501     22.4
502     20.6
503     23.9
504     22.0
505     11.9

```

Name: medv, Length: 501, dtype: float64

```
[ ]: x_train, x_test, y_train, y_test = train_test_split(x, y ,test_size = 0.2,
↳random_state=0)
```

```
[ ]: from sklearn.linear_model import LinearRegression
      regressor = LinearRegression()
```

```
[ ]: regressor.fit(x_train, y_train)
```

```
[ ]: LinearRegression()
```

```
[ ]: y_pred = regressor.predict(x_test)
```

```
[ ]: from sklearn.metrics import mean_squared_error, mean_absolute_error
      mse = mean_squared_error(y_test, y_pred)
```

```

mae = mean_absolute_error(y_test,y_pred)
print("Mean Square Error : ", mse)
print("Mean Absolute Error : ", mae)

```

Mean Square Error : 41.056168987190816  
Mean Absolute Error : 3.941158173657429

```

[ ]: print(mean_absolute_error(y_test, y_pred))
      print(mean_squared_error(y_test, y_pred))
      print(num.sqrt(mse))

```

3.941158173657429  
41.056168987190816  
6.407508797277677

```

[ ]: plt.scatter(y_test, y_pred, c = 'green')
      plt.xlabel("Price: in $1000's")
      plt.ylabel("Predicted value")
      plt.title("True value vs predicted value : Linear Regression")
      plt.show()

```

