Experiment-7

7) Consider the Boston dataset and train Multiple Linear Regression and evaluate the model using different matrices

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
```

In [2]:
```
##Multiple Linear Regression using Boston house price prediction
```

In [3]:
```
df=pd.read_csv('C:/Users/Guru Kiran/All CSV files/Boston.csv')
df.head(5)
```

Out[3]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  506 non-null    int64  
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64  
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64  
 10  tax         506 non-null    int64  
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

In [5]:
```
x = df.drop('medv',axis=1)
y = df['medv']
```

In [6]:
```
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, randc
```

In [7]:
```
## Build the model
```

In [8]:
```
lr = LinearRegression()
model = lr.fit(x_train,y_train)
y_pred = model.predict(x_test)
```

In [9]:
```
y_pred = model.predict(x_test)
```

Evaluation metrics

In [10]:
```
# Metrics
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("MSE:", mse)
print("RMSE:", rmse)
print("R² Score:", r2)
```

```
MSE: 21.176142408242043
RMSE: 4.6017542750827145
R² Score: 0.7295300033236576
```