

```
#Akshata Jadhav
#TE - A - 24
#Batch TA1-TA2
```

✓ Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import StandardScaler

boston = load_boston()
boston.keys()

dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])

x = pd.DataFrame(boston.data, columns=boston.feature_names)
y = pd.DataFrame(boston.target, columns=['MEDV'])
```

```
x.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	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
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.93
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
x.shape, y.shape
```

```
((506, 13), (506, 1))
```

✓ Basic stats

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 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    float64
4    NOX         506 non-null    float64
5    RM          506 non-null    float64
6    AGE         506 non-null    float64
7    DIS         506 non-null    float64
8    RAD         506 non-null    float64
9    TAX         506 non-null    float64
10   PTRATIO     506 non-null    float64
11   B           506 non-null    float64
12   LSTAT       506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

```
x.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

```
y.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 1 columns):
#   Column  Non-Null Count  Dtype
---  -
0   MEDV    506 non-null      float64
dtypes: float64(1)
memory usage: 4.1 KB
```

```
y.describe()

MEDV
count 506.000000
mean 22.532806
std 9.197104
min 5.000000
25% 17.025000
50% 21.200000
75% 25.000000
max 50.000000
```

```
x.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
dtype: int64
```

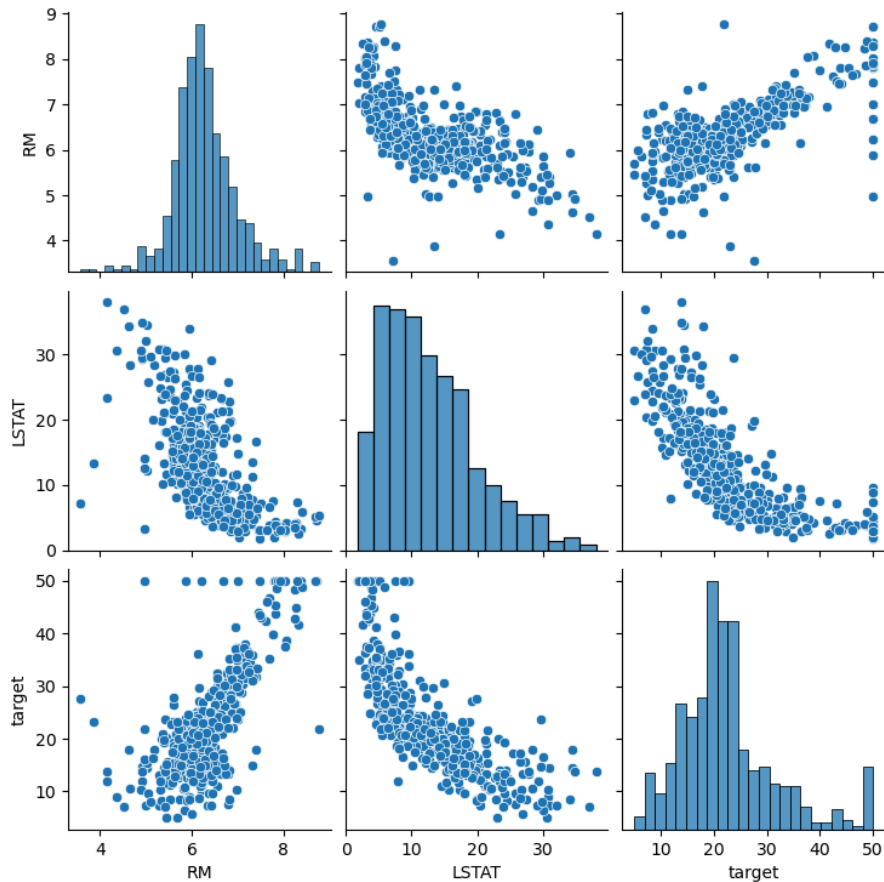
```
y.isnull().sum()

MEDV 0
dtype: int64
```

```
df = x
df["target"] = y
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.96
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.35

```
sns.pairplot(df)
plt.show()
```



✓ Scale the data

```
scaler = StandardScaler()
```

```
x = scaler.fit_transform(x)
```

✓ Split the data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, shuffle=True)
```

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((354, 2), (152, 2), (354,), (152,))
```

✓ Linear Regression Modelling

```
model = LinearRegression(n_jobs=-1)

model.fit(x_train, y_train)

LinearRegression(n_jobs=-1)
```

✓ Make predictions

```
y_pred = model.predict(x_test)

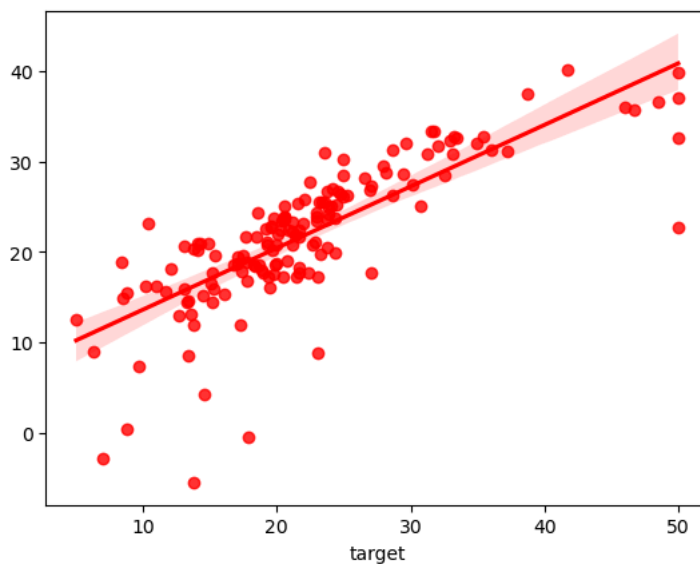
mean_absolute_error(y_test, y_pred)

3.701010266760501

mean_squared_error(y_test, y_pred)

30.5001478179898

sns.regplot(y_test, y_pred, color='red')
plt.show()
```



Start coding or [generate](#) with AI.