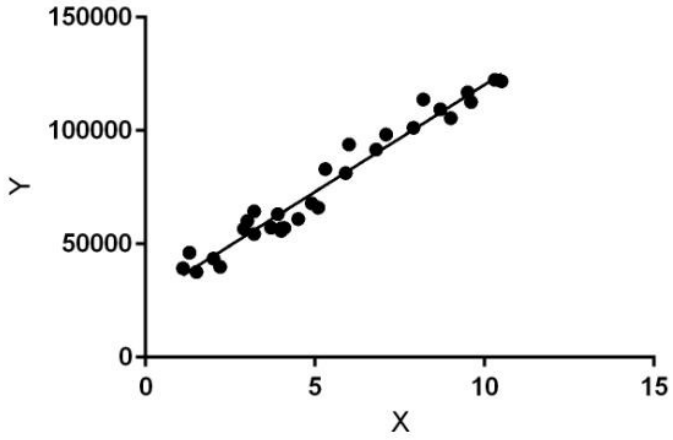
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

Aim: Analyze the Boston Housing dataset and apply appropriate Regression Technique.

Objective: Ablility to perform various feature engineering tasks, apply linear regression on the given dataset and minimise the error.

Theory:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Dataset:

The Boston Housing Dataset

The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:

CRIM - per capita crime rate by town

ZN - proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS - proportion of non-retail business acres per town.

CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX - nitric oxides concentration (parts per 10 million)

RM - average number of rooms per dwelling

AGE - proportion of owner-occupied units built prior to 1940

DIS - weighted distances to five Boston employment centres

RAD - index of accessibility to radial highways

TAX - full-value property-tax rate per $10,000

PTRATIO - pupil-teacher ratio by town

B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT - % lower status of the population

MEDV - Median value of owner-occupied homes in $1000's

Conclusion:

1. Features chosen to develop the model are : LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' ,'RAD' , 'AGE' , 'CRIM' . These features were chosen because they have high correlation with the target feature : MEDV. LSTAT', 'RM', and 'PTRATIO' are recognized predictors of housing prices due to their intuitive impact on demand and desirability. The additional incorporation of 'INDUS', 'TAX', 'NOX', 'RAD', 'AGE', and 'CRIM' offers insights into socio-economic and environmental factors influencing housing values. Considering these features enhances your predictive model's ability to capture nuances, potentially leading to more accurate 'MEDV' predictions.

2. Mean Squared Error (MSE) assesses predictive model accuracy by calculating the average squared difference between predicted and actual values. It emphasizes larger deviations, with lower MSE indicating better performance in reducing prediction errors. Mean squared error for the given problem is calculated as : 29.52

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv("/content/drive/MyDrive/dataset/HousingData.csv")

df.head()
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX
PTRATIO  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296
15.3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242
17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242
17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222
18.7
4  0.06905   0.0   2.18   0.0  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    NaN  36.2
```

```
df.describe().T
```

```
         count        mean         std        min         25%
50%  \
CRIM     486.0    3.611874    8.720192    0.00632    0.081900
0.253715
ZN       486.0   11.211934   23.388876    0.00000    0.000000
0.000000
INDUS    486.0   11.083992    6.835896    0.46000    5.190000
9.690000
CHAS     486.0    0.069959    0.255340    0.00000    0.000000
0.000000
NOX      506.0    0.554695    0.115878    0.38500    0.449000
0.538000
RM       506.0    6.284634    0.702617    3.56100    5.885500
6.208500
AGE      486.0   68.518519   27.999513    2.90000   45.175000
```

```
76.800000
DIS      506.0    3.795043    2.105710    1.12960    2.100175
3.207450
RAD      506.0    9.549407    8.707259    1.00000    4.000000
5.000000
TAX      506.0  408.237154  168.537116  187.00000  279.000000
330.000000
PTRATIO  506.0   18.455534    2.164946   12.60000   17.400000
19.050000
B        506.0  356.674032   91.294864    0.32000  375.377500
391.440000
LSTAT    486.0   12.715432    7.155871    1.73000    7.125000
11.430000
MEDV     506.0   22.532806    9.197104    5.00000   17.025000
21.200000

                 75%       max
CRIM        3.560263   88.9762
ZN         12.500000  100.0000
INDUS      18.100000   27.7400
CHAS        0.000000    1.0000
NOX         0.624000    0.8710
RM          6.623500    8.7800
AGE        93.975000  100.0000
DIS         5.188425   12.1265
RAD        24.000000   24.0000
TAX       666.000000  711.0000
PTRATIO    20.200000   22.0000
B         396.225000  396.9000
LSTAT      16.955000   37.9700
MEDV       25.000000   50.0000

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 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    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

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

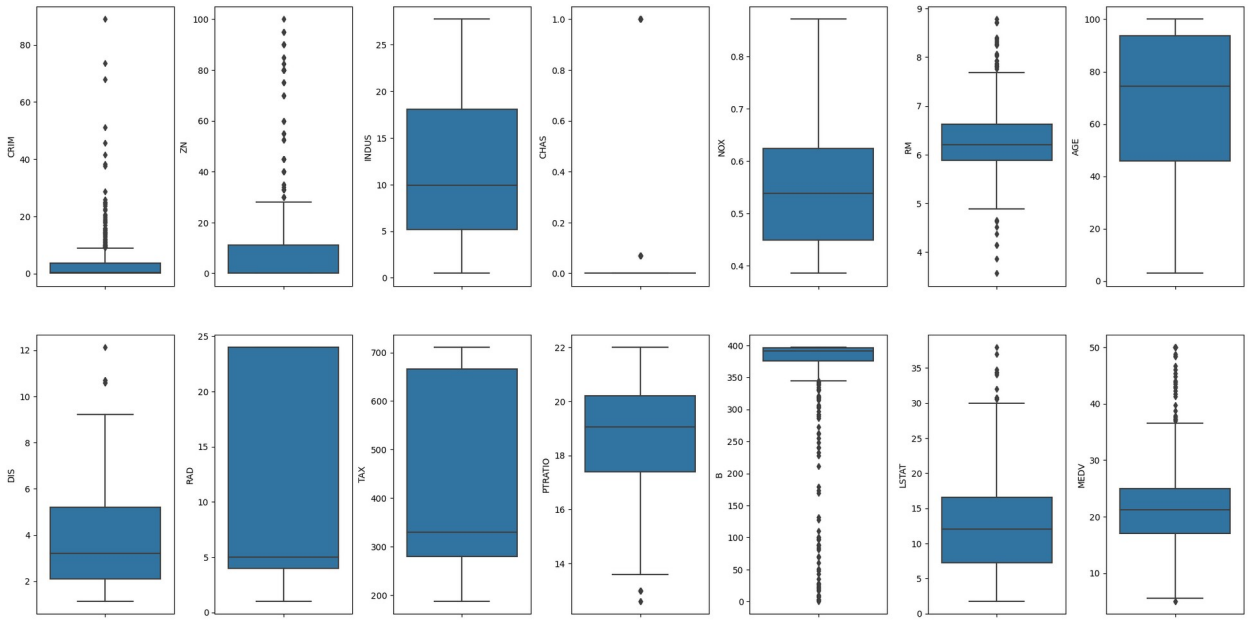
```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

```
df['CRIM'].fillna(df['CRIM'].mean() , inplace = True)

df['ZN'].fillna(df['ZN'].mean() , inplace = True)
df['INDUS'].fillna(df['INDUS'].mean() , inplace = True)
df['CHAS'].fillna(df['CHAS'].mean() , inplace = True)
df['AGE'].fillna(df['AGE'].mean() , inplace = True)
df['LSTAT'].fillna(df['LSTAT'].mean() , inplace = True)
```

```
df.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
MEDV       0
dtype: int64
```

```
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
ax = ax.flatten()

for col, value in df.items():
    sns.boxplot(y=col, data=df, ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```




```
fig, ax = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
ax = ax.flatten()

for col, value in df.items():
    sns.distplot(value, ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```

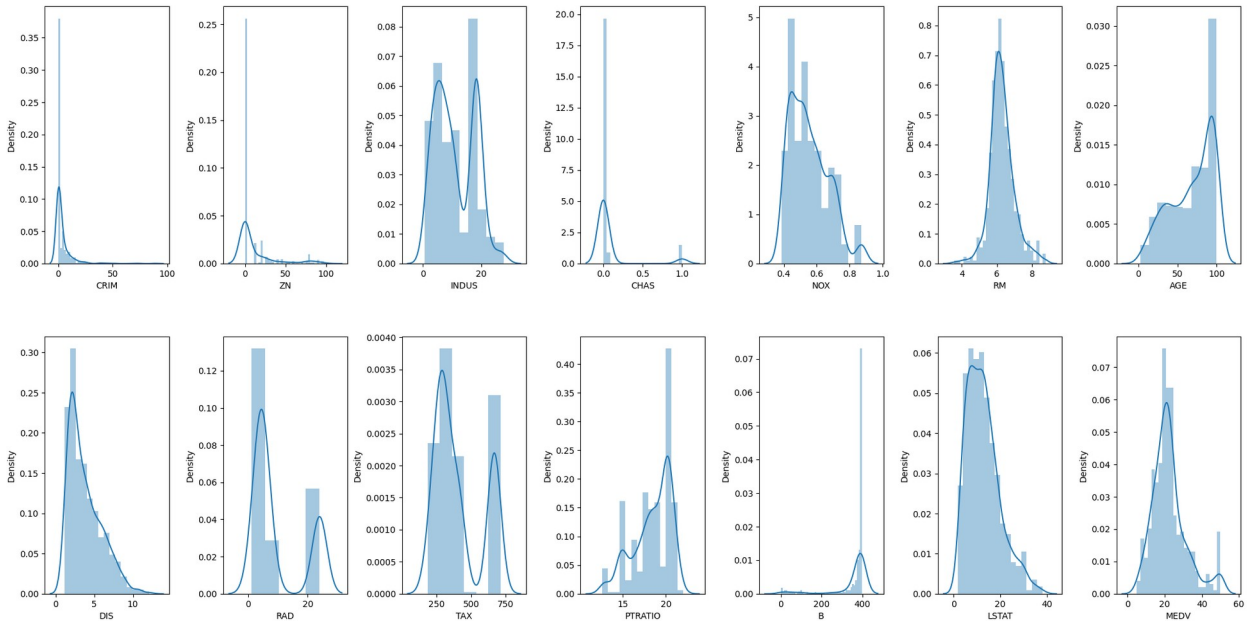
```
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
```

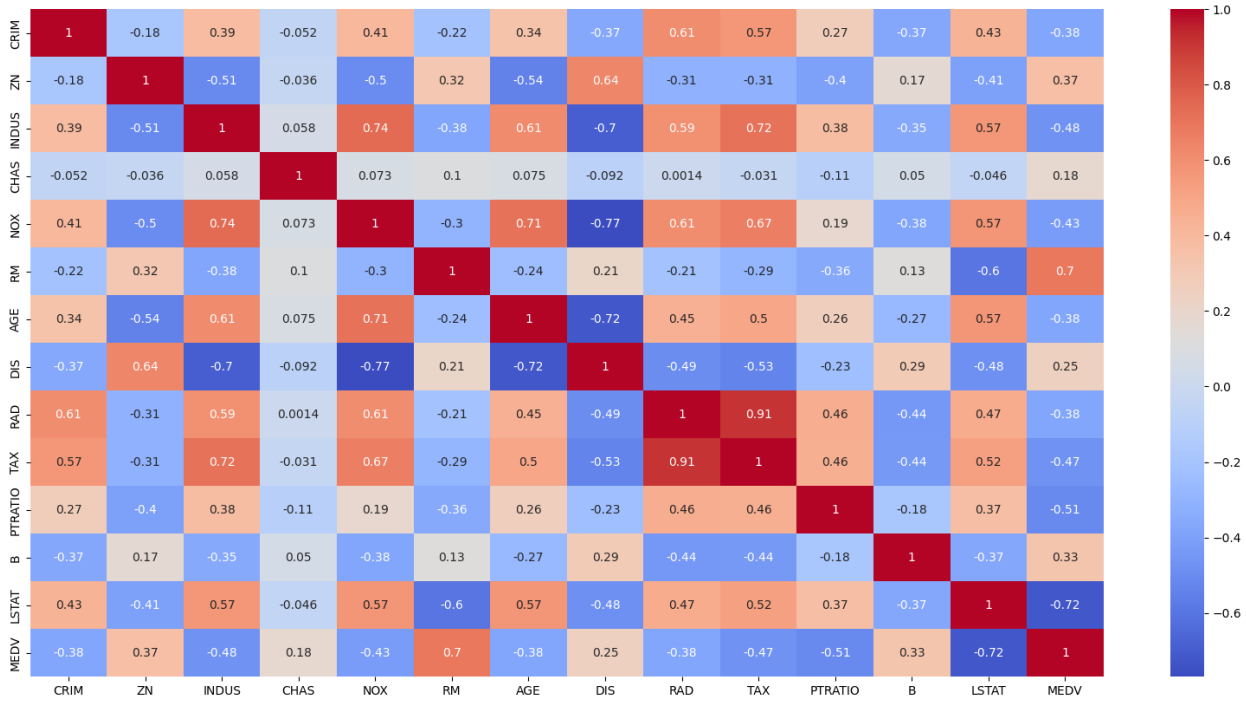
```
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
```

```
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
<ipython-input-12-433c1c8df893>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(value, ax=ax[index])
```



```
corr = df.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot=True, cmap='coolwarm')

<Axes: >
```



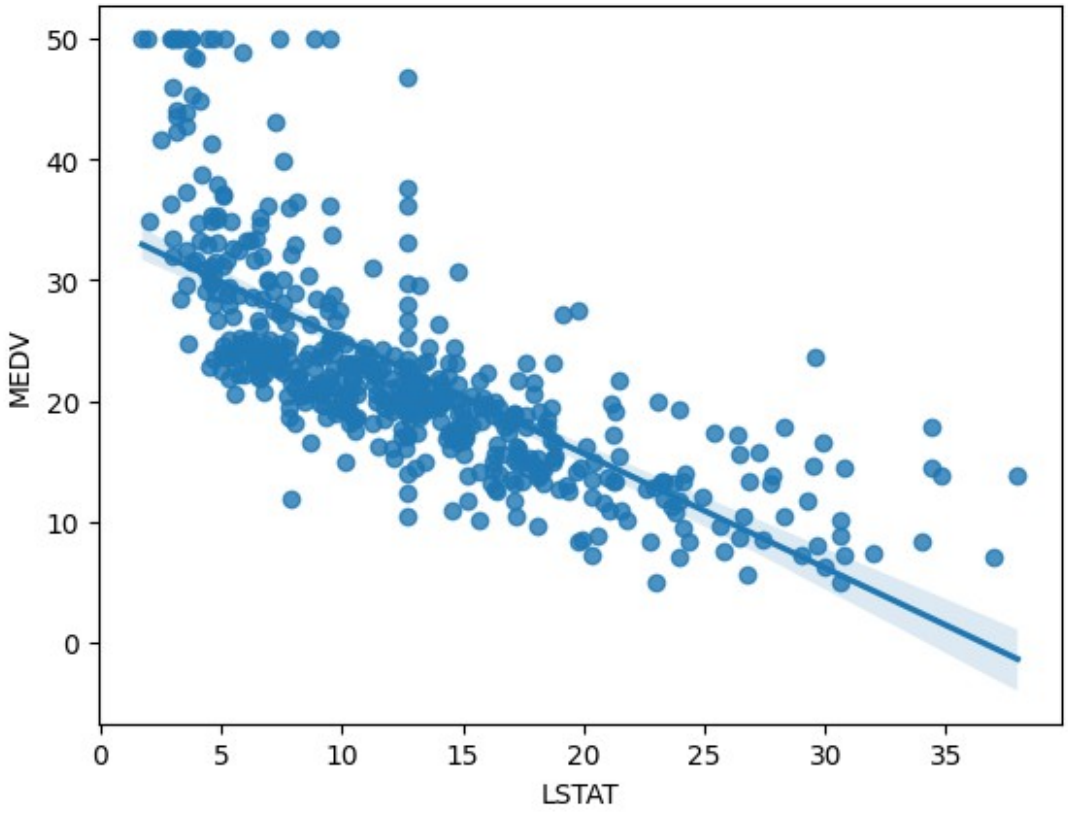

```
df.corr()['MEDV'].values

array([-0.37969547,  0.36594312, -0.47865733,  0.1798825 , -
0.42732077,
        0.69535995, -0.38022344,  0.24992873, -0.38162623, -
0.46853593,
       -0.50778669,  0.33346082, -0.72197464,  1.        ])

df.corr()['MEDV'].index

Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')

corr_values = df.corr()['MEDV']

sorted_corr = corr_values.abs().sort_values(ascending= False)

sorted_features = sorted_corr.index

sorted_correlation_df = pd.DataFrame({'Feature': sorted_features,
'Correlation': sorted_corr})
```

```
sorted_correlation_df
```

	Feature	Correlation
MEDV	MEDV	1.000000
LSTAT	LSTAT	0.721975
RM	RM	0.695360
PTRATIO	PTRATIO	0.507787
INDUS	INDUS	0.478657
TAX	TAX	0.468536
NOX	NOX	0.427321
RAD	RAD	0.381626
AGE	AGE	0.380223
CRIM	CRIM	0.379695
ZN	ZN	0.365943
B	B	0.333461
DIS	DIS	0.249929
CHAS	CHAS	0.179882

```
sns.regplot(y=df['MEDV'], x=df['LSTAT'])
```

```
<Axes: xlabel='LSTAT', ylabel='MEDV'>
```




```
from sklearn.model_selection import cross_val_score, train_test_split
```

```
sorted_correlation_df
```

```
         Feature  Correlation
MEDV        MEDV     1.000000
LSTAT      LSTAT     0.721975
RM            RM     0.695360
PTRATIO  PTRATIO     0.507787
INDUS      INDUS     0.478657
TAX          TAX     0.468536
NOX          NOX     0.427321
RAD          RAD     0.381626
AGE          AGE     0.380223
CRIM        CRIM     0.379695
ZN             ZN     0.365943
B               B     0.333461
DIS          DIS     0.249929
CHAS        CHAS     0.179882
```

```
sorted_correlation_df[:10]
```

```
         Feature  Correlation
MEDV        MEDV     1.000000
LSTAT      LSTAT     0.721975
RM            RM     0.695360
PTRATIO  PTRATIO     0.507787
INDUS      INDUS     0.478657
TAX          TAX     0.468536
NOX          NOX     0.427321
RAD          RAD     0.381626
AGE          AGE     0.380223
CRIM        CRIM     0.379695
```

```
sorted_correlation_df[:11]['Feature'].index
```

```
Index(['MEDV', 'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX', 'RAD',
'AGE',
       'CRIM', 'ZN'],
      dtype='object')
```

```
df.head()
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX
PTRATIO  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296
15.3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242
17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242
17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222
18.7
```

```
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222
18.7

        B      LSTAT  MEDV
0  396.90   4.980000  24.0
1  396.90   9.140000  21.6
2  392.83   4.030000  34.7
3  394.63   2.940000  33.4
4  396.90  12.715432  36.2
```

```
X = df[['LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' , 'RAD' ,
'AGE' , 'CRIM' , 'ZN']]
Y = df['MEDV']
seed= 1
X_train , X_test, Y_train , Y_test = train_test_split(X, Y,
test_size=0.20, random_state=seed)
```

```
X.shape
```

```
(506, 10)
```

```
Y.shape
```

```
(506,)
```

```
from sklearn.linear_model import LinearRegression
```

```
LR=LinearRegression()
```

```
LR.fit(X_train , Y_train)
```

```
LinearRegression()
```

```
ypred= LR.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error , mean_absolute_error
```

```
mae=mean_absolute_error(Y_test , ypred)
print(mae)
```

```
4.311333848096257
```

```
mse=mean_squared_error(Y_test , ypred)
print(mse)
```

```
29.58597268132346
```

```
me=np.sqrt(mse)
print(me)
```

```
5.439298914503914
```

Feature Considered and the error observed :

** All Features : MAE = 3.88 , MSE=36.12 , ME=6.01

** 'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' : MAE = 4.23 , MSE=30.9 , ME=5.56

** 'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' ,'RAD' : MAE = 4.26 , MSE=29.65 , ME=5.44

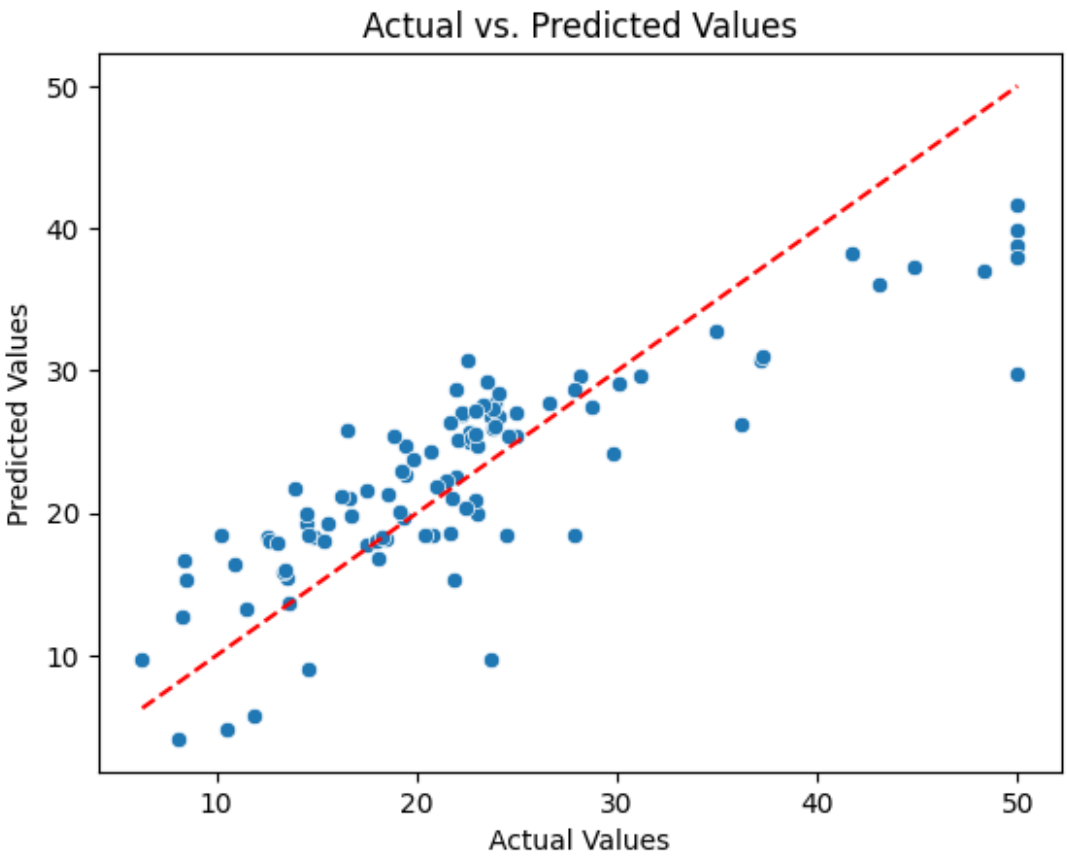
** 'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' ,'RAD' , 'AGE': MAE = 4.28 , MSE=29.53 , ME=5.43

'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' ,'RAD' , 'AGE' , 'CRIM': MAE = 4.31 , MSE=29.52 , ME=5.43

** 'LSTAT', 'RM', 'PTRATIO', 'INDUS', 'TAX', 'NOX' ,'RAD' , 'AGE' , 'CRIM' , 'ZN': MAE = 4.31 , MSE=29.58 , ME=5.43

Actual V/S Predicted

```
data = pd.DataFrame({'Actual': Y_test , 'Predicted': ypred})

# Create a scatter plot
sns.scatterplot(x='Actual', y='Predicted', data=data)

# Add a diagonal line for reference (perfect prediction)
plt.plot([min(data['Actual']), max(data['Actual'])],
[min(data['Actual']), max(data['Actual'])], color='red',
linestyle='--')

plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```


Actual vs. Predicted Values
Predicted Values
Actual Values
10
20
30
40
50