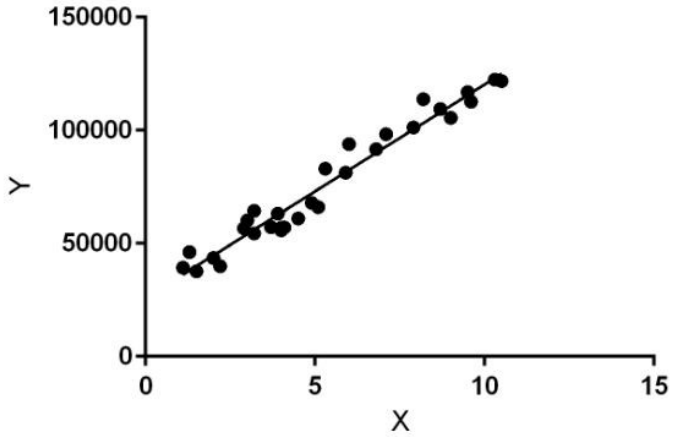
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance: 24-07-2023
Date of Submission: 11-09-2023

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

Code:

Conclusion:

1. Feature used:
   chosen features are logical candidates for predicting house prices. For example, features like "RM" (average number of rooms) and "LSTAT" (% lower status of the population) are often indicative of the quality and socioeconomic status of the neighborhood, which can influence housing prices. "TAX" and "PTRATIO" might reflect the quality of local schools, which can also impact housing prices. Similarly, features like "NOX" (nitric oxides concentration) and "DIS" (distance to employment centers) could influence the desirability of an area and therefore affect house prices.

2. **MSA** : Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

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

```
df = pd.read_csv("HousingData.csv")
```

```
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	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN

```
df.shape
```

```
(506, 14)
```

```
df.columns
```

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

```
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
```

Checking for Null Values

```
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
```

Remove All Null Values

```
df = df.dropna()
```

```
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
```

Train-Test Split

```
from sklearn.model_selection import train_test_split
```

```
x = x=df.drop('MEDV',axis=1)
y=df.MEDV
```

```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.30,random_state=42)
```

```
x_train
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
21	0.85204	0.0	8.14	0.0	0.538	5.965	89.2	4.0123	4	307	21.0	392.53	13.8
456	4.66883	0.0	18.10	0.0	0.713	5.976	87.9	2.5806	24	666	20.2	10.48	19.0
28	0.77299	0.0	8.14	0.0	0.538	6.495	94.4	4.4547	4	307	21.0	387.94	12.8
156	2.44668	0.0	19.58	0.0	0.871	5.272	94.0	1.7364	5	403	14.7	88.63	16.1
445	10.67180	0.0	18.10	0.0	0.740	6.459	94.8	1.9879	24	666	20.2	43.06	23.9
...	...	...	...	...	...	...	...	...	...	...	...	...	.
84	0.05059	0.0	4.49	0.0	0.449	6.389	48.0	4.7794	3	247	18.5	396.90	9.6
128	0.32543	0.0	21.89	0.0	0.624	6.431	98.8	1.8125	4	437	21.2	396.90	15.3
345	0.03113	0.0	4.39	0.0	0.442	6.014	48.5	8.0136	3	352	18.8	385.64	10.5
448	9.32909	0.0	18.10	0.0	0.713	6.185	98.7	2.2616	24	666	20.2	396.90	18.1
122	0.09299	0.0	25.65	0.0	0.581	5.961	92.9	2.0869	2	188	19.1	378.09	17.9

275 rows × 13 columns

```
x_test
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST
93	0.02875	28.0	15.04	0.0	0.464	6.211	28.9	3.6659	4	270	18.2	396.33	6.
352	0.07244	60.0	1.69	0.0	0.411	5.884	18.5	10.7103	4	411	18.3	392.33	7.
319	0.47547	0.0	9.90	0.0	0.544	6.113	58.8	4.0019	4	304	18.4	396.23	12.
65	0.03584	80.0	3.37	0.0	0.398	6.290	17.8	6.6115	4	337	16.1	396.90	4.
499	0.17783	0.0	9.69	0.0	0.585	5.569	73.5	2.3999	6	391	19.2	395.77	15.
...	...	...	...	...	...	...	...	...	...	...	...	...	
358	5.20177	0.0	18.10	1.0	0.770	6.127	83.4	2.7227	24	666	20.2	395.43	11.
457	8.20058	0.0	18.10	0.0	0.713	5.936	80.3	2.7792	24	666	20.2	3.50	16.
321	0.18159	0.0	7.38	0.0	0.493	6.376	54.3	4.5404	5	287	19.6	396.90	6.
190	0.09068	45.0	3.44	0.0	0.437	6.951	21.5	6.4798	5	398	15.2	377.68	5.

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

```
x_train=scaler.fit_transform(x_train)
```

```
x_test=scaler.transform(x_test)
```

Model Training

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

```
r=LinearRegression(fit_intercept=True, copy_X=True, n_jobs=None, positive=False)
```

```
r.fit(x_train,y_train)
```

▾ LinearRegression
LinearRegression()

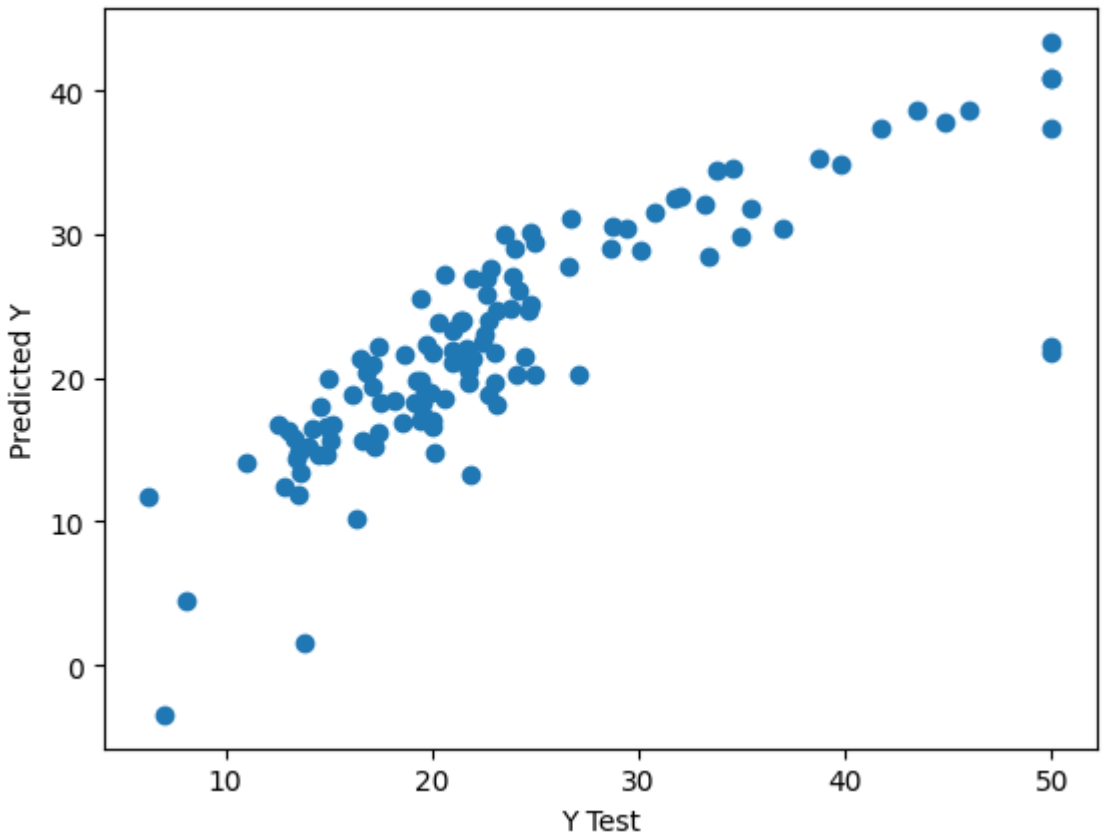
```
predictions = r.predict(x_test)
```

Model Performance

```
plt.scatter(y_test,predictions)
```

```
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Text(0, 0.5, 'Predicted Y')
```




```
from sklearn import metrics


print('MAE:',metrics.mean_squared_error(y_test,predictions))
print('MSE:',metrics.mean_absolute_error(y_test,predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

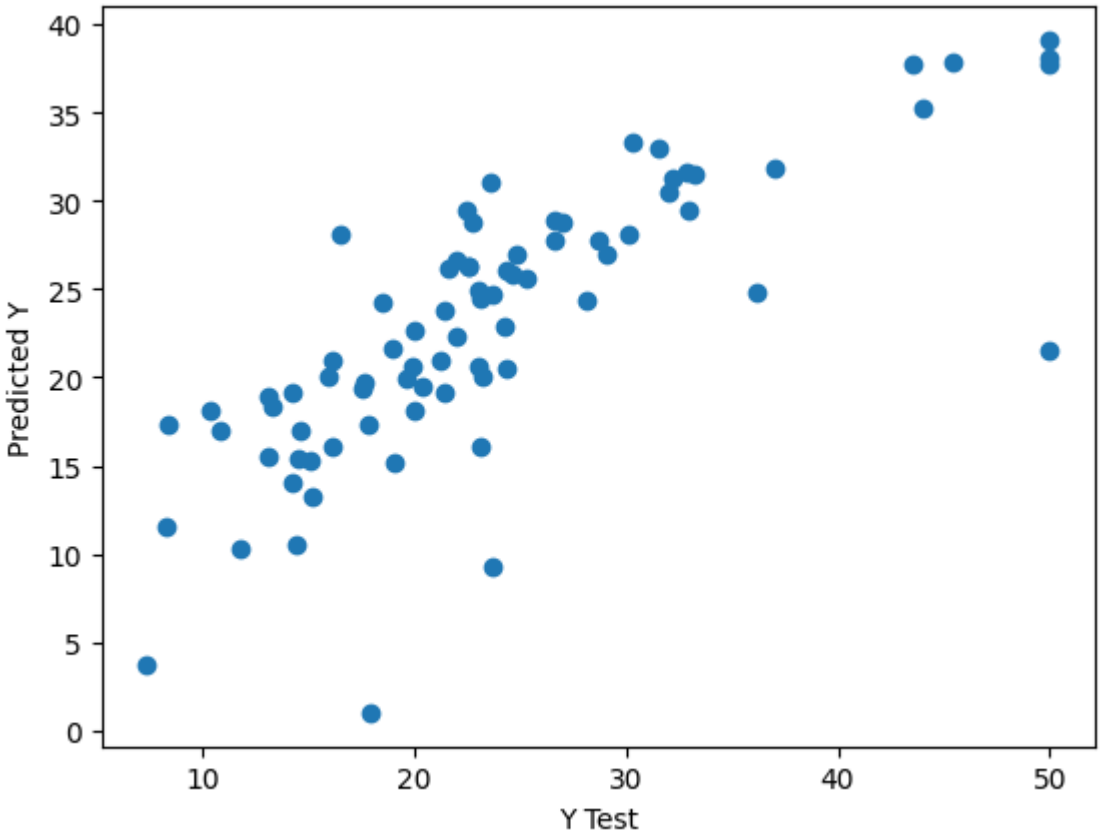
```
MAE: 28.870771928253443
MSE: 3.4558210072479936
RMSE: 5.373152885248422
```

```
x=df[['CRIM','INDUS','RM','AGE','TAX','PTRATIO','LSTAT']]
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2)


r=LinearRegression()
r.fit(X_train,y_train)
predictions = r.predict(X_test)


plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Text(0, 0.5, 'Predicted Y')
```




```
print('MAE:',metrics.mean_squared_error(y_test,predictions))
print('MSE:',metrics.mean_absolute_error(y_test,predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE: 37.82588304119944
MSE: 4.2149479769193485
RMSE: 6.150275037849888
```