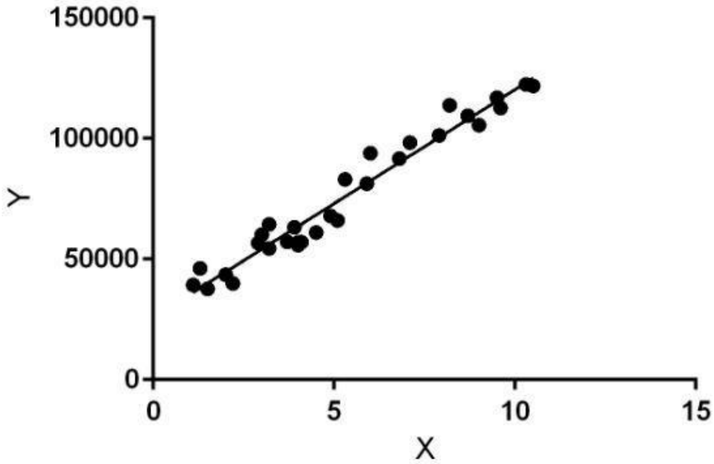
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

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

Code:

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

```
dataset=pd.read_csv('Boston-house-price-data.csv')
X=dataset.iloc[:,:-1].values
Y=dataset.iloc[:,-1].values
```

```
dataset.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.1
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.0
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.9
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.3

+ Code + Text

```
dataset.shape
```

```
(506, 14)
```

```
dataset.info()
```

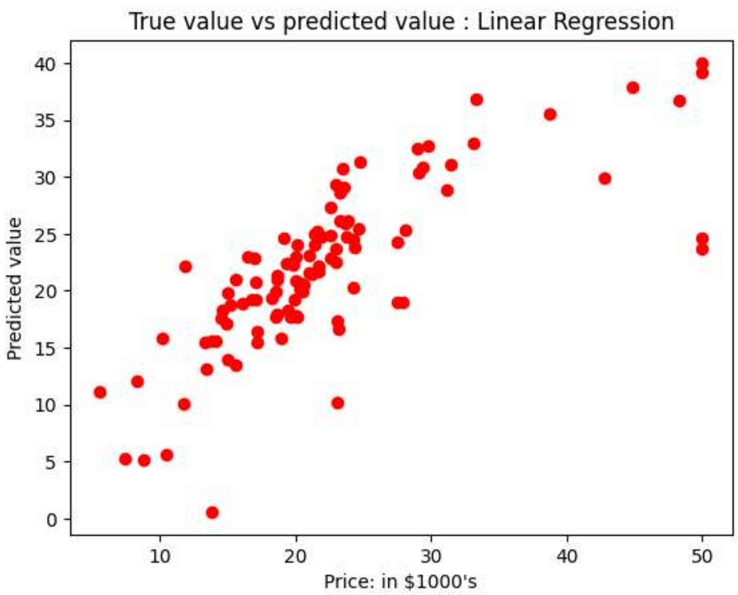
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS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

```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values

from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,random_state = 0)

print("xtrain shape : ", xtrain.shape)
print("xtest shape  : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape  : ", ytest.shape)
```

```
xtrain shape :  (404, 13)
xtest shape  :  (102, 13)
ytrain shape :  (404,)
ytest shape  :  (102,)
```

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

y_pred = regressor.predict(xtest)
```

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

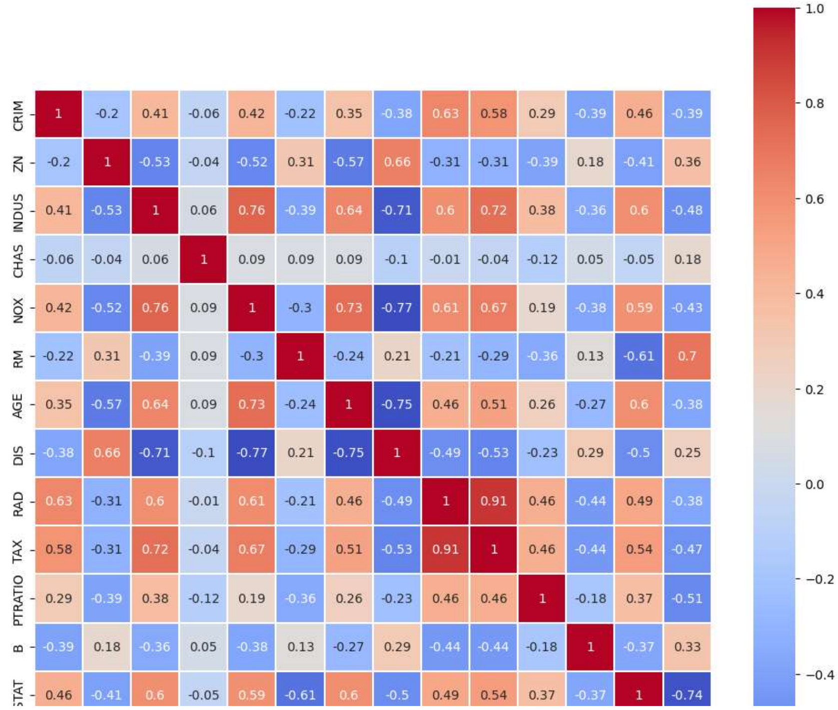



```
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(ytest, y_pred)
mae = mean_absolute_error(ytest,y_pred)
print("Mean Square Error : ", mse)
print("Mean Absolute Error : ", mae)
```

```
Mean Square Error :  33.44897999767653
Mean Absolute Error :  3.8429092204444966
```

```
import seaborn as sns
plt.figure(figsize=(12,12))
sns.heatmap(data=dataset.corr().round(2),annot=True,cmap='coolwarm',linewidths=0.2,square=True)
```

```
<Axes: >
```




```
df1 = dataset[['RM','TAX','PTRATIO','LSTAT']]
df1.head()
```

	RM	TAX	PTRATIO	LSTAT
0	6.575	296.0	15.3	4.98
1	6.421	242.0	17.8	9.14
2	7.185	242.0	17.8	4.03
3	6.998	222.0	18.7	2.94
4	7.147	222.0	18.7	5.33

```
df1.shape
```

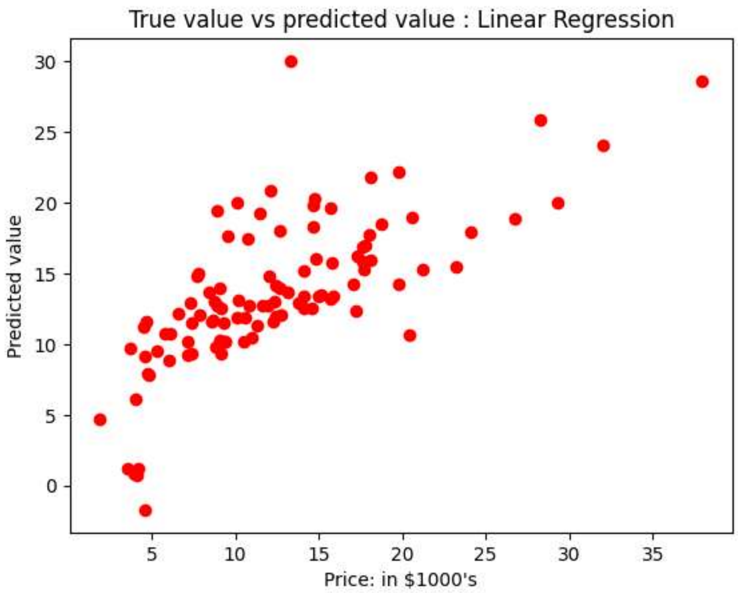
```
(506, 4)
```

```
x=df1.iloc[:,:-1].values
y=df1.iloc[:,-1].values

from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,random_state = 0)

print("xtrain shape : ", xtrain.shape)
print("xtest shape  : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape  : ", ytest.shape)
```

```
xtrain shape :  (404, 3)
xtest shape  :  (102, 3)
ytrain shape :  (404,)
ytest shape  :  (102,)
```

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(xtrain, ytrain)

y_pred = regressor.predict(xtest)
```

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




```
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(ytest, y_pred)
mae = mean_absolute_error(ytest,y_pred)
print("Mean Square Error : ", mse)
print("Mean Absolute Error : ", mae)
# ,'MEDV' removed end maI
```

```
Mean Square Error :  21.714680825959494
Mean Absolute Error :  3.616572667697949
```

Conclusion:

We have used the following features in our dataset to predict the values of houses

1. CRIM - per capita crime rate by town shows safety affects the prices
2. ZN - this feature shows spacious plots which affects the house price
3. INDUS - the proportion of retail business affects the prices as it provides ease to customers
4. CHAS - Location near a river can increase the house price as it is an attraction
5. NOX - Pollution affects the house prices as no one wants to live in polluted area
6. RM - No of room increases the prices
7. AGE - Older houses can increase price due to significant architecture or decrease due to being old
8. DIS - Being located near employment centers can significantly affect house prices
9. RAD - Location near highway provides ease of transport hence contributing significantly over prices
10. TAX - Taxes provides significant roles in house prices
11. PTRATIO - Lower ratio indicates more quality of education hence contributing to house prices.
12. B - This ratio may prove to be significant due to social-economic reasons.
13. LSTAT - Higher ratio may lead to poor population contributing to house prices
14. MEDV - This feature may also prove to be significant in house prices

The calculated Mean Squared Error (MSE) value is approximately 33.6213. MSE measures the average squared difference between the predicted and actual values. In the context of housing prices, this value indicates the average squared difference between the model's predicted prices and the actual prices of houses in the test set.