

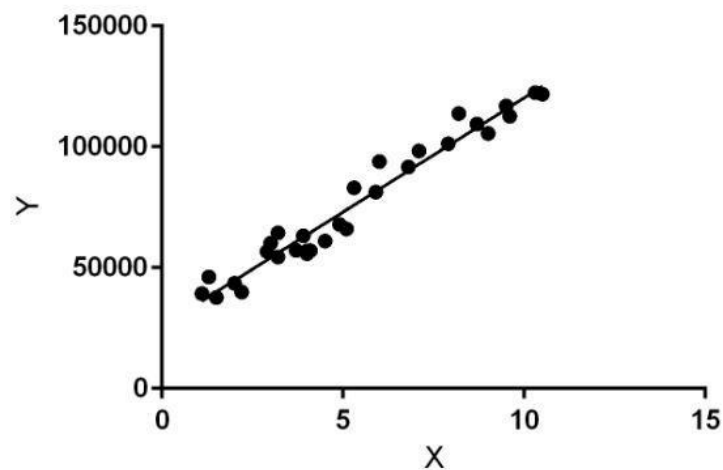
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance: 24/07/2023
Date of Submission: 10/8/2023

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

Objective: Ability 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 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 - $1000(B_k - 0.63)^2$ where B_k 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:

Here we learn about linear regression which can be used to predict the any aspect of the database. Linear Regression helps in predicting the values using mathematical calculations.

The considered points are:

'CRIM' , 'ZN' , 'INDUS' , 'CHAS' , 'AGE' , 'DIS' , 'RAD' , 'TAX' , 'PTRATIO' , 'B' , 'LSTAT'

The Mean Square Error is : 28.70878219455333

The Absolute Error is : 3.639210759954872

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

```
data=pd.read_csv('Boston-house-price-data.csv')
```

```
data.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

```
new_ds=data[['CRIM', 'ZN', 'INDUS', 'CHAS', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']]
```

```
print("Shape of dataset:", data.shape)
```

```
Shape of dataset: (506, 14)
```

```
data.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=new_ds.iloc[:, :-1].values
y=new_ds.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)
```

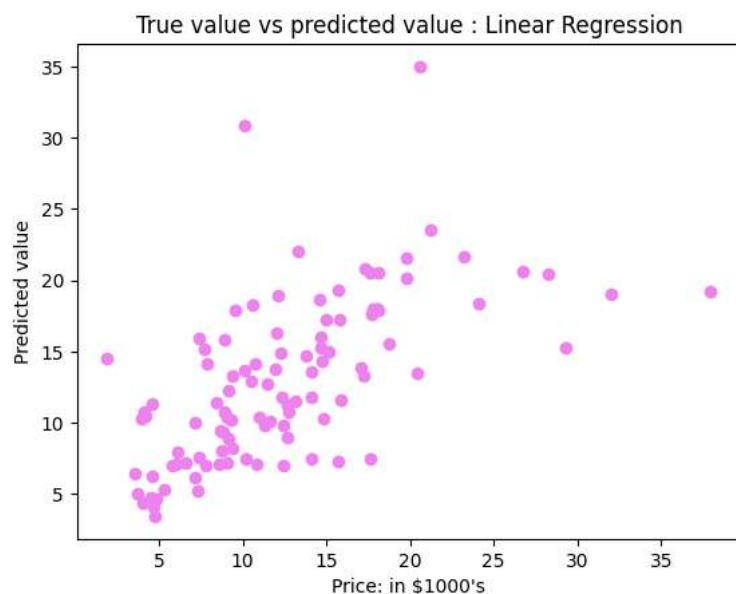
```
xtrain shape : (404, 10)
xtest shape : (102, 10)
ytrain shape : (404,)
ytest shape : (102,)
```

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

```
y_pred = regressor.predict(xtest)
```

```
import matplotlib.pyplot as plt
```

```
plt.scatter(ytest, y_pred, c = 'violet')  
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 : 28.70878219455333  
Mean Absolute Error : 3.6392107599548726
```

✓ 0s completed at 00:21

