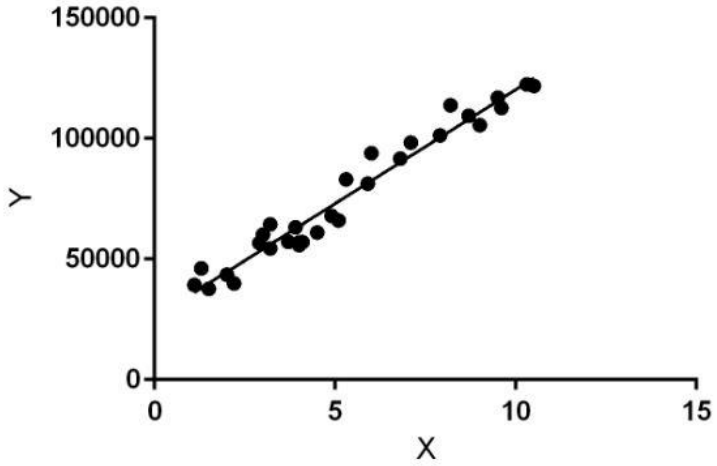
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:

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:

Conclusion:

1. What are features have been chosen to develop the model? Justify the features chosen to estimate the price of a house.

   In this experiment we learned the concept of Linear regression which is used to predict the relationship between a dependent variable and one or more independent variables using mathematical calculations.

The points considered are :
‘CRIM’,’ZN’,’INDUS’,’CHAS’,’AGE’,’DIS’,’RAD’,’TAX’,’PTRATIO’,’B’,’LSTAT

2. Comment on the Mean Squared Error calculated.

   The Mean Squared Error(MSE) measures the amount of error in statistical models. It measures the average squared difference between the actual values and the predicted values produced by the regression model.MSE is used to evaluate the quality of the model’s predictions and to compare different models’ performance.

   I considered all the features and the Mean Squared Error is : 35.011

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

```
dataset = pd.read_csv('/content/BostonHousing.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
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90

```
dataset.zn.replace(0,np.nan,inplace=True)
dataset.chas.replace(0,np.nan,inplace=True)
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       134 non-null    float64
 2   indus    506 non-null    float64
 3   chas     35 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    int64  
 9   tax      506 non-null    int64  
 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
```

```
dataset.shape
```

```
(506, 12)
```

```
#calculate the percentage of null values
dataset.isnull().sum()/len(dataset) * 100
```

```
crim        0.000000
zn         73.517787
indus       0.000000
chas       93.083004
nox         0.000000
rm          0.000000
age         0.000000
```

```
dis         0.000000
rad         0.000000
tax         0.000000
ptratio     0.000000
b           0.000000
lstat       0.000000
medv        0.000000
dtype: float64
```

```
#dropping the columns
dataset = dataset.drop(['zn','chas'],axis=1)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 12 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   indus    506 non-null    float64
 2   nox      506 non-null    float64
 3   rm       506 non-null    float64
 4   age      506 non-null    float64
 5   dis      506 non-null    float64
 6   rad      506 non-null    int64  
 7   tax      506 non-null    int64  
 8   ptratio  506 non-null    float64
 9   b        506 non-null    float64
 10  lstat    506 non-null    float64
 11  medv     506 non-null    float64
dtypes: float64(10), int64(2)
memory usage: 47.6 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)
```

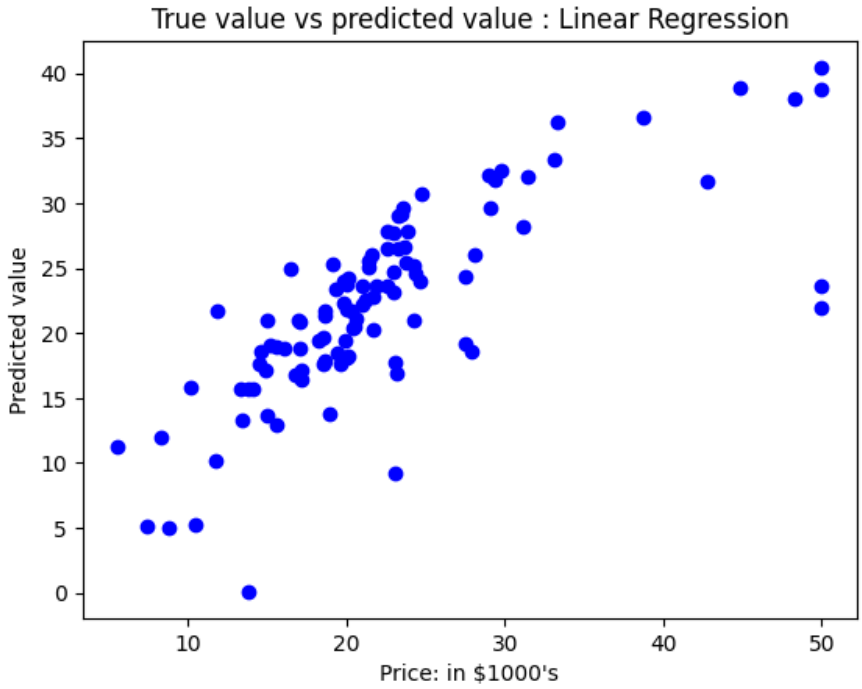
```
xtrain shape :  (404, 11)
xtest shape  :  (102, 11)
ytrain shape :  (404,)
ytest shape  :  (102,)
```

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

y_pred = regressor.predict(xtest)



plt.scatter(ytest, y_pred, c = 'blue')
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 :  35.01170873305253
Mean Absolute Error :  3.949685793659988
```