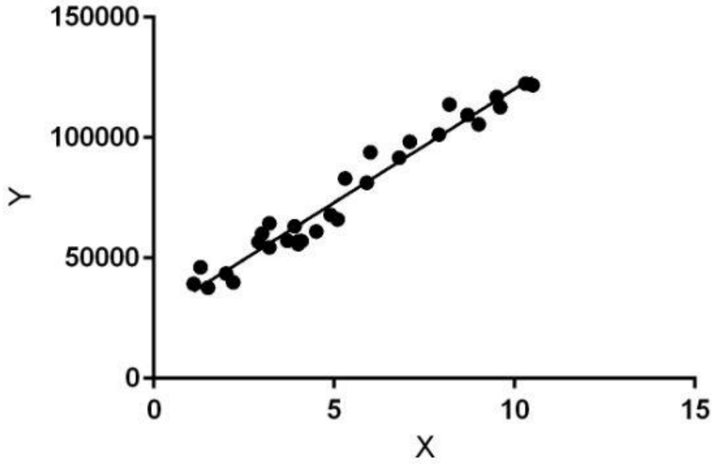
Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:24-07-2023
Date of Submission:11-08-2023

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

Conclusion:

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

CRIM: The crime rate in a neighborhood can affect the desirability of an area. Higher crime rates may lead to lower property values due to safety concerns.

ZN: Zoning for larger residential lots can be indicative of upscale neighborhoods, influencing higher property values.

INDUS: A higher proportion of non-retail business land may indicate less residential character, potentially affecting property values.

CHAS: The proximity of a property to a river might be seen as an attractive feature, potentially increasing its value.

NOX: Nitric oxide concentration can be a measure of air pollution. Higher pollution levels might lead to decreased property values.

RM: The average number of rooms can reflect the size of the dwelling. Larger homes tend to have higher prices.

AGE: The age of owner-occupied units might be correlated with maintenance and modernity. Older properties might have lower values.

DIS: Proximity to employment centers can increase property values due to convenience for workers.

RAD: Better accessibility to highways and transportation can positively impact property values.

TAX: Higher property taxes might lower a property's appeal and affordability, affecting its value.

PTRATIO: A lower pupil-teacher ratio might be desirable for families, potentially affecting property values.

B: The proportion of Black residents in a neighborhood might be a socio-economic indicator that could impact property values.

LSTAT: A higher percentage of lower-status population might reflect the economic conditions of the area and impact property values.

2. Comment on the Mean Squared Error calculated.

   The Mean Absolute Error (MAE) assesses regression model performance. MAE measures average absolute differences between predicted and actual values. Lower MAE signifies closer predictions to actual values. In training, MAE of 0.0146 indicates accurate model predictions. In testing, MAE of 2.2835 shows reasonably accurate predictions but potential overfitting, as it's larger than training MAE. Balancing accuracy on both datasets can enhance model's generalization.

HOUSE PRICE PREDICTION

Importing dependencies

```
import numpy as np
import pandas as pd
from pandas import read_csv
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

Importing the dataset

```
dataset =read_csv("/content/boston.csv")
```

```
dataset.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

```
print(np.shape(dataset))
```

```
(506, 14)
```

```
dataset.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
```

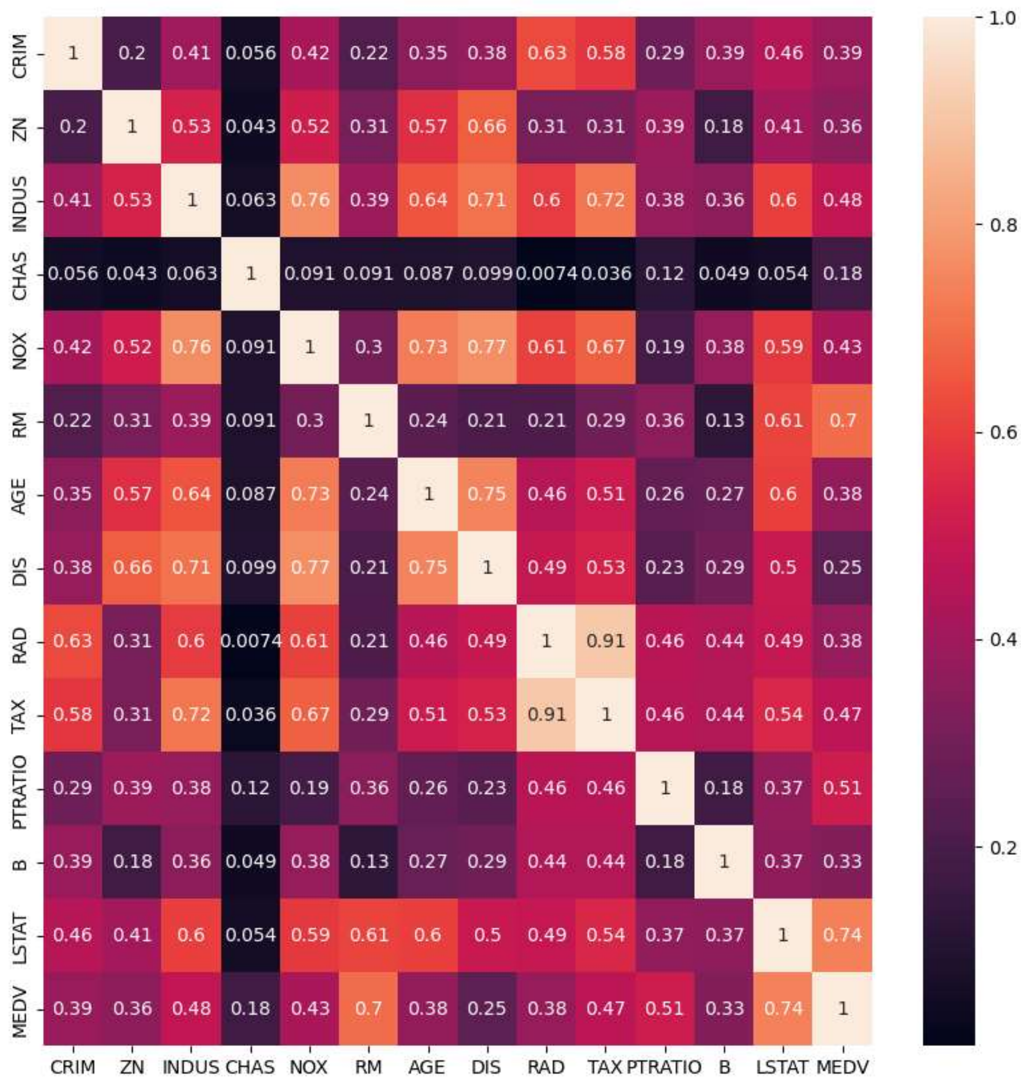
```
dataset.describe()
```

```
correlation = dataset.corr()
```

```
      count   506.000000   506.000000   506.000000   506.000000   506.000000   506.000000   506.000000   506.000000   506.
```

```
plt.figure(figsize=(10,10))
sns.heatmap(dataset.corr().abs(),annot = True)
```

```
<Axes: >
```




```
X = dataset.drop(['MEDV'],axis = 1)
Y = dataset['MEDV']
```

```
print(X)
```

```
        CRIM    ZN  INDUS  CHAS    NOX  ...  RAD    TAX  PTRATIO       B  LSTAT
0    0.00632  18.0   2.31     0  0.538  ...    1  296.0     15.3  396.90   4.98
1    0.02731   0.0   7.07     0  0.469  ...    2  242.0     17.8  396.90   9.14
2    0.02729   0.0   7.07     0  0.469  ...    2  242.0     17.8  392.83   4.03
3    0.03237   0.0   2.18     0  0.458  ...    3  222.0     18.7  394.63   2.94
4    0.06905   0.0   2.18     0  0.458  ...    3  222.0     18.7  396.90   5.33
..       ...   ...    ...   ...    ...  ...  ...    ...      ...     ...    ...
501  0.06263   0.0  11.93     0  0.573  ...    1  273.0     21.0  391.99   9.67
502  0.04527   0.0  11.93     0  0.573  ...    1  273.0     21.0  396.90   9.08
503  0.06076   0.0  11.93     0  0.573  ...    1  273.0     21.0  396.90   5.64
504  0.10959   0.0  11.93     0  0.573  ...    1  273.0     21.0  393.45   6.48
505  0.04741   0.0  11.93     0  0.573  ...    1  273.0     21.0  396.90   7.88

[506 rows x 13 columns]
```

```
print(Y)
```

```
0      24.0
1      21.6
```

```
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

Splitting

```
X_train,X_test,Y_train,Y_test  = train_test_split(X,Y,test_size =0.2,random_state = 2)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(506, 13) (404, 13) (102, 13)
```

model training

```
model = XGBRegressor()
```

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

```
▾                              XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

```
prediction = model.predict(X_train)
```

```
print(prediction)
```

```
[23.147501  20.99463   20.090284  34.69053   13.903663  13.510157
 21.998634  15.1940975 10.899711  22.709627  13.832816   5.592794
 29.810236  49.99096   34.89215   20.607384  23.351097  19.23555
 32.695698  19.641418  26.991022   8.401829  46.00729   21.708961
 27.062933  19.321356  19.288303  24.809872  22.61626   31.70493
 18.542515   8.697379  17.395294  23.700663  13.304856  10.492197
 12.688369  25.016556  19.67495   14.902088  24.193798  25.007143
 14.900281  16.995798  15.6009035 12.699232  24.51537   14.999952
 50.00104   17.525454  21.184624  31.998049  15.613355  22.89754
 19.325378  18.717896  23.301125  37.222923  30.09486   33.102703
 21.00072   49.999332  13.405827   5.0280113 16.492886   8.405072
 28.64328   19.499939  20.586452  45.402164  39.79833   33.407326
 19.83506   33.406372  25.271482  50.001534  12.521657  17.457413
 18.61758   22.602625  50.002117  23.801117  23.317268  23.087355
 41.700035  16.119293  31.620516  36.069206   7.0022025 20.3827
 19.996452  11.986318  25.023014  49.970123  37.881588  23.123034
 41.292133  17.596548  16.305374  30.034231  22.860699  19.810343
 17.098848  18.898268  18.96717   22.606049  23.141363  33.183487
 15.010934  11.693824  18.78828   20.80524   17.99983   19.68991
 50.00332   17.207317  16.404053  17.520426  14.593481  33.110855
 14.508482  43.821655  34.939106  20.381636  14.655634   8.094332
 11.7662115 11.846876  18.69599    6.314154  23.983706  13.084503
 19.603905  49.989143  22.300608  18.930315  31.197134  20.69645
 32.21111   36.15102   14.240763  15.698188  49.99381   20.423601
 16.184978  13.409128  50.01321   31.602146  12.271495  19.219482
 29.794909  31.536846  22.798779  10.189648  24.08648   23.710463
 21.991894  13.802495  28.420696  33.181534  13.105958  18.988266
 26.576572  36.967175  30.794083  22.77071   10.201246  22.213818
 24.483162  36.178806  23.09194   20.097307  19.470194  10.786644
 22.671095  19.502405  20.109184   9.611871  42.799637  48.794792
 13.097208  20.28583   24.793974  14.110478  21.701134  22.217012
 33.003544  21.11041   25.00658   19.122992  32.398567  13.605098
 15.1145315 23.088867  27.474783  19.364998  26.487135  27.499458
```

```
28.697094  21.21718   18.703201  26.775208  14.010719  21.692347
18.372562  43.11582   29.081839  20.289959  23.680176  18.308306
17.204844  18.320065  24.393475  26.396057  19.094141  13.3019905
22.15311   22.185797   8.516214  18.894428  21.792608  19.331121
18.197924   7.5006843 22.406403  20.004215  14.412416  22.503702
28.53306   21.591028  13.810223  20.497831  21.898977  23.104464
49.99585   16.242056  30.294561  50.001595  17.771557  19.053703
10.399217  20.378187  16.49973   17.183376  16.70228   19.495337
30.507633  28.98067   19.528809  23.148346  24.391027   9.521643
23.886024  49.995125  21.167099  22.597813  19.965279  13.4072275
19.948694  17.087479  12.738807  23.00453   15.222122  20.604322
26.207253  18.09243   24.090246  14.105     21.689667  20.08065
25.010437  27.874954  22.92366   18.509727  22.190847  24.004797
14.788686  19.89675   24.39812   17.796036  24.556297  31.970308
17.774675  23.356768  16.134794  13.009915  10.98219   24.28906
15.56895   35.209793  19.605724  42.301712   8.797891  24.400295
14.086652  15.408639  17.301126  22.127419  23.09363   44.79579
17.776684  31.50014   22.835577  16.888603  23.925127  12.097476
38.685944  21.388391  15.98878   23.912495  11.909485  24.960499
 7.2018585 24.696215  18.201897  22.489008  23.03332   24.260433
17.101519  17.805563  13.493165  27.105328  13.311978  21.913465
20.00738   15.405392  16.595737  22.301016  24.708412  21.422579
22.878702  29.606575  21.877811  19.900253  29.605219  23.407152
13.781474  24.454706  11.897682   7.2203646 20.521074   9.725295
48.30087   25.19501   11.688618  17.404732  14.480284  28.618876
```

R squared error

```
score1 = metrics.r2_score(Y_train,prediction)
```

MAE

```
score2 = metrics.mean_absolute_error(Y_train,prediction)
```

```
print('R2',score1)
```

```
R2 0.9999948236320982
```

```
print('MAE',score2)
```

```
MAE 0.0145848437110976
```

```
prediction_test = model.predict(X_test)
```

```
score1 = metrics.r2_score(Y_test,prediction_test)
```

```
score2 = metrics.mean_absolute_error(Y_test,prediction_test)
```

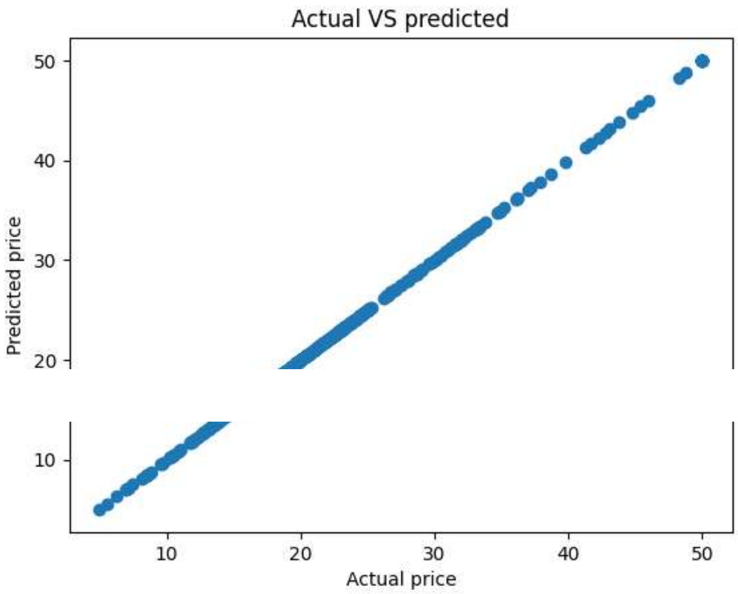
```
print('R2',score1)
```

```
R2 0.8711660369151691
```

```
print('MAE',score2)
```

```
MAE 2.2834744154238233
```

```
plt.scatter(Y_train,prediction)
plt.xlabel("Actual price")
plt.ylabel("Predicted price")
plt.title("Actual VS predicted")
plt.show()
```


Actual VS predicted
Predicted price
Actual price
10
20
30
40
50
10
20
30
40
50