1. Impoting Dependencies :

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

2. Data Collcetion and Analysis:

A- Loading dataset of Boston House Price:

In [ ]:
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing
col_names = ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "T
data = pd.read_csv(url, delim_whitespace=True, names=col_names)
```

B- View the data (head)

In [ ]:
```
data.head()
```

Out[ ]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.!
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.!
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.!
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.(
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.!

C-Number of row & columns:

In [ ]:
```
data.shape
```

Out[ ]: (506, 14)

D- Check for missing values;

In [ ]:
```
data.isnull().sum()
#We don't have any missing valie
```

Out[ ]:
```
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
```

2. Statisctical measures :

A. General Statistic:

In [ ]:
```
data.describe()
```

Out[ ]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	A
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574(
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148(
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900(
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500(
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075(
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000(

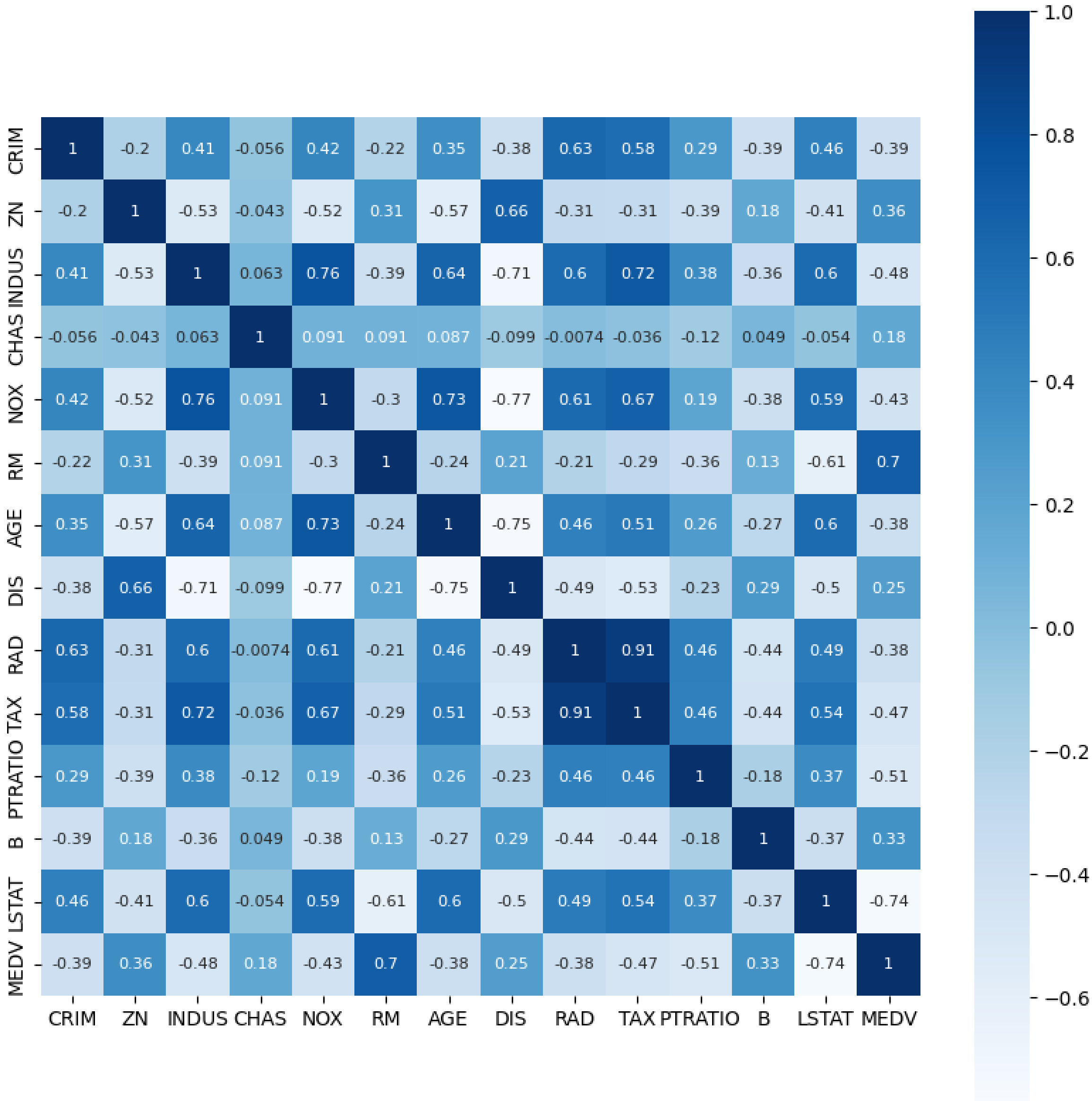
3. Correlation between various features in the dataset:

In [ ]:
```
correlation = data.corr()
```

A. Constructing a heatmap to understand the correlation

In [ ]:
```
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, annot=True, annot_kws={'size':8
```

Out[ ]:
```
<Axes: >
```



4. Spilitti,g the data and Target (Price):

A- Separating the data and labels

In [ ]:
```
X = data.drop(['MEDV'], axis= 1)
Y = data['MEDV']
print(X)
print(Y)
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273.0
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273.0
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273.0
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273.0
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273.0

     PTRATIO       B  LSTAT
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90   5.33
..       ...     ...    ...
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

[506 rows x 13 columns]
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
```

B- Spiliting the data into Training & test data:

In [ ]:
```python
X_train, X_test,Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_st
```

In [ ]:
```python
print(X.shape,X_train.shape, X_test.shape)
```

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

5. Model Training:

A- Loading the model

In [ ]:
```python
model = XGBRegressor()
```

B- Training the model:

In [ ]:
```python
model.fit(X_train,Y_train)
```

Out[ ]:

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_roun
ds=None,
             enable_categorical=False, eval_metric=None, feature_typ
es=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_b
in=None,
```

6. Evaluation of model:

A. Prediction on training data:

In [ ]:

```
data_prediction = model.predict(X_train)
print(data_prediction)
```

```
[23.112196  20.992601  20.10438   34.67932   13.920501  13.499354
 21.998383  15.206723  10.89543   22.67402   13.795236   5.602332
 29.808502  49.98666   34.89634   20.594336  23.388903  19.2118
 32.69294   19.604128  26.978151   8.405952  46.00062   21.70406
 27.084402  19.372278  19.297894  24.79984   22.608278  31.707775
 18.53683    8.703393  17.40025   23.698814  13.29729   10.504759
 12.693588  24.994888  19.694864  14.911037  24.20254   24.991112
 14.901547  16.987965  15.592753  12.704759  24.505623  15.007718
 49.999355  17.509344  21.18844   31.999287  15.606071  22.902134
 19.309835  18.697083  23.302961  37.19767   30.102247  33.117855
 20.993683  50.00471   13.40048    5.002565  16.50862    8.4016905
 28.651423  19.49218   20.595366  45.404697  39.808857  33.4055
 19.81498   33.406376  25.30206   49.998615  12.544487  17.433802
 18.602612  22.601418  50.004013  23.814182  23.313164  23.097467
 41.71243   16.112017  31.604454  36.09397    7.0009975 20.406271
 19.992195  12.003392  25.027754  49.98552   37.890903  23.091173
 41.289513  17.604618  16.30125   30.05175   22.884857  19.802671
 17.106977  18.903633  18.897047  22.598665  23.170893  33.19197
 15.00434   11.704804  18.795511  20.817484  17.998543  19.633396
 49.998672  17.208574  16.410513  17.506626  14.6008    33.09849
 14.504811  43.813366  34.900055  20.388191  14.605566   8.091776
 11.777508  11.811628  18.691      6.322443  23.97163   13.073076
 19.595     49.99033   22.319597  18.91175   31.203646  20.712711
 32.200443  36.188755  14.222898  15.705663  50.000664  20.408077
 16.185907  13.410434  50.012474  31.60327   12.288182  19.18906
 29.809902  31.49241   22.804003  10.194443  24.09609   23.705154
 22.008154  13.790835  28.399841  33.199585  13.102867  19.017357
 26.61559   36.963135  30.7939    22.80785   10.206419  22.19713
 24.482466  36.19345   23.092129  20.12124   19.498154  10.796299
 22.701403  19.49908   20.107922   9.625605  42.797676  48.79655
 13.099009  20.29537   24.794712  14.106459  21.698246  22.188694
 32.99889   21.09952   24.998121  19.110165  32.401157  13.601795
 15.072056  23.06062   27.487326  19.401924  26.481848  27.50343
 28.686726  21.19214   18.701029  26.7093    14.01264   21.699009
 18.39739   43.11556   29.09378   20.298742  23.711458  18.30434
 17.193619  18.321108  24.392206  26.391497  19.10248   13.302614
 22.189732  22.199099   8.530714  18.889635  21.800455  19.305798
 18.198288   7.4938145 22.400797  20.028303  14.404203  22.500402
 28.504164  21.608568  13.798578  20.495127  21.902288  23.100073
 50.00128   16.23443   30.298399  49.996014  17.78638   19.060133
 10.39715   20.383387  16.496948  17.195917  16.681927  19.509869
 30.502445  29.01701   19.558786  23.172018  24.397314   9.528121
 23.894762  49.996834  21.196695  22.596247  19.989746  13.393513
 19.995872  17.068512  12.718964  23.01111   15.199219  20.609226
 26.19055   18.109114  24.098877  14.100204  21.695303  20.096022
 25.018776  27.899471  22.918222  18.499252  22.202477  23.99494
 14.8048935 19.896328  24.411158  17.790047  24.596226  32.007046
 17.778685  23.309103  16.120615  13.003008  10.993355  24.306978
 15.597863  35.20248   19.58716   42.29605    8.789314  24.399925
 14.109244  15.4010315 17.299047  22.113592  23.106049  44.805172
 17.795519  31.499706  22.813938  16.836212  23.911596  12.09551
 38.69628   21.387049  16.001123  23.929094  11.897898  24.983562
  7.1969633 24.69086   18.187803  22.471941  23.013317  24.295506
 17.099222  17.796907  13.503164  27.094381  13.296886  21.90404
 19.99361   15.402385  16.588629  22.29326   24.697983  21.428938
 22.882269  29.601665  21.881992  19.908726  29.60596   23.408524
 13.807421  24.499699  11.901903   7.20547   20.484905   9.706262
 48.301437  25.194635  11.691466  17.39672   14.49594   28.584557
 19.395731  22.486904   7.0219784 20.60076   22.998001  19.699215
 23.700571  25.02278   27.992222  13.39496   14.524017  20.30391
```

```
19.304321  24.108646  14.88511   26.387497  33.31608   23.61982
24.60193   18.494753  20.90211   10.411172  23.305649  13.097067
24.699335  22.610847  20.50208   16.82098   10.198874  33.805454
18.60289   50.0009    23.778967  23.91014   21.15922   18.81689
 8.491747  21.506403  23.200815  21.043766  16.604784  28.060492
21.197857  28.370916  14.2918625 49.997353  30.989647  24.980095
21.410505  19.000553  29.00484   15.204052  22.791481  21.791014
19.896528  23.77255  ]
```

B- R squared erreor

In [ ]:
```
score_1 = metrics.r2_score(Y_train, data_prediction)
```

C- Mean Absolute Error

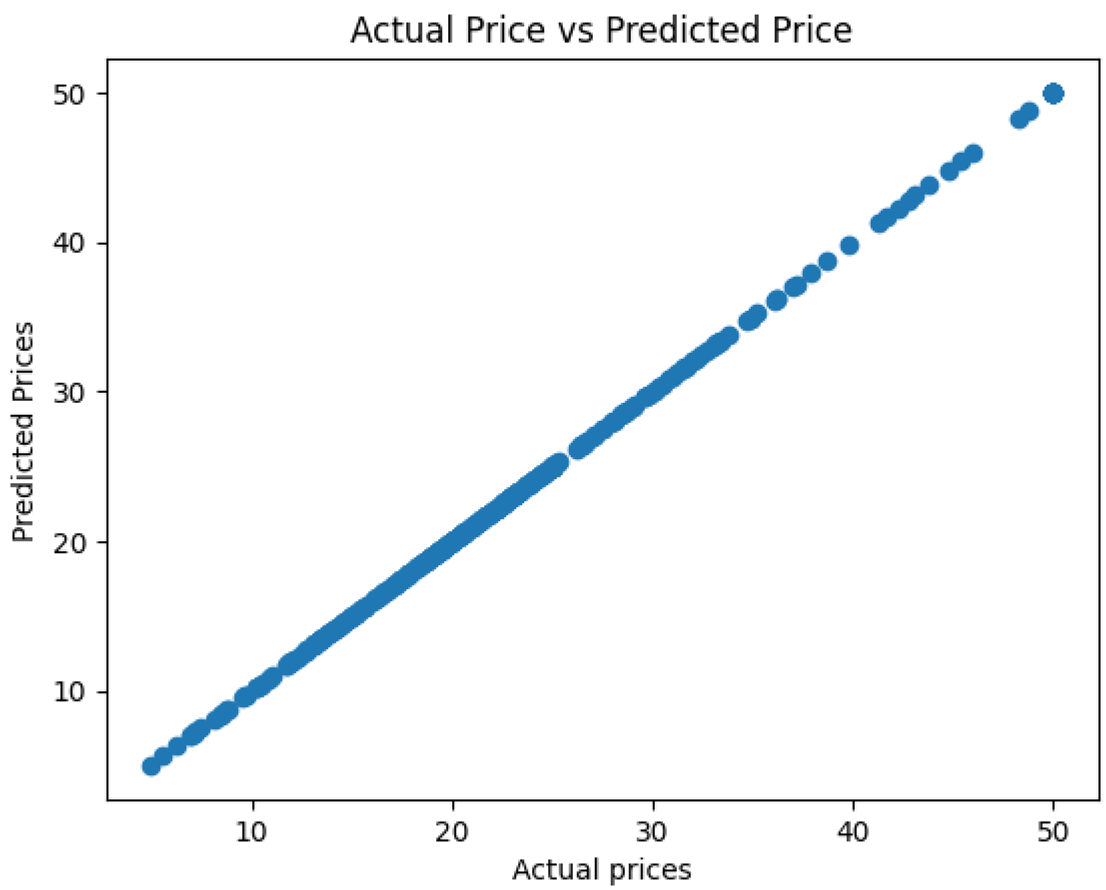
In [ ]:
```
score_2 = metrics.mean_absolute_error(Y_train, data_prediction)
```

In [ ]:
```
print("R quared error :", score_1)
print("Mean Absolute Error :", score_2)
```

```
R quared error : 0.9999980039471451
Mean Absolute Error : 0.0091330346494618
```

D- Visualisation the actual Prices and Predict prices:

In [ ]:
```
plt.scatter(Y_train, data_prediction)
plt.xlabel('Actual prices')
plt.ylabel("Predicted Prices")
plt.title("Actual Price vs Predicted Price")
plt.show()
```



E. Prediction on testing data:

In [ ]:
```
data_prediction_testing= model.predict(X_test)
print(data_prediction_testing)
```

```
[22.007828  21.22598   30.466019  27.735027   9.134951  12.740403
 25.738058  27.750889  25.364376  20.229292  27.821787  24.7761
 19.771252  20.497349  12.970438  22.86288   19.605635  10.677987
  8.277654  15.529657  22.842052  20.002996  34.06762   18.943192
 15.624948  18.787666  46.0246    33.05114   34.804283  19.070232
 17.53711   20.27066   31.102339  24.026129  12.199101  18.224184
 10.182956  21.252314  22.891352  21.458113  26.451164  12.1898775
 27.141438   8.322471  21.356699  12.768549  35.221687  14.574406
 32.06173   15.088605  31.076805  26.808199   6.1558666 34.42615
 25.135347  19.508772  19.424906  19.58183   16.680052  22.962534
 20.904106  21.24      18.46788   29.243906  33.434864  26.021257
 49.91979   25.905489   9.713634  24.058743  16.63922    9.0341625
 13.197622  18.80479   26.985659  24.746912  22.200838  21.017391
 19.30188   24.098715  34.517494  19.51518   20.331131  31.346212
 47.815742  36.102997  17.42751   24.595816  29.387545  18.68302
 19.893139  20.184433  11.331679  38.306778  42.119137   9.208766
 43.026043  34.444504  21.611591  17.832836  27.724092  23.295132 ]
```

In [ ]:
```
score_1test = metrics.r2_score(Y_test, data_prediction_testing)
score_2test = metrics.mean_absolute_error(Y_test, data_prediction_testing)
print("R quared error :", score_1test)
print("Mean Absolute Error :", score_2test)
```

```
R quared error : 0.9051721149855378
Mean Absolute Error : 2.0748727686264927
```