In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
%matplotlib inline
```

In [2]:
```python
hp2= sklearn.datasets.load_boston()
print(hp2)
```


```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
         4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
         9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
         4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
         6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         7.8800e+00]]), 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
       15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
       17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
       25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
       23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
       32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
       34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
       20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
       26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
       31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
       22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
       42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
       36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
       32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
       20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
       20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
       22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
       21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
       19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
       32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
       18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
       16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
       13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
        7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
       12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
       27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
        8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
        9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
       10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]), 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'), 'DESCR': ".. _boston_dataset:\n\nBoston house prices dataset\n---------------------------\n\n**Data Set Characteristics:**  \n\n    :Number of Instances: 506 \n\n    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.\n\n    :Attribute Information (in order):\n        - CRIM     per capita crime rate by town\n        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.\n        - INDUS    proportion of non-retail business acres per town\n        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\n        - NOX      nitric oxides concentration (parts per 10 million)\n        - RM       average number of rooms per dwelling\n        - AGE      proportion of owner-occupied units built prior to 1940\n        - DIS      weighted distances to five Boston employment centres\n        - RAD      index of accessibility to radial highways\n        - TAX      full-value property-tax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by town\n        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black people by town\n        - LSTAT    % lower status of the population\n        - MEDV     Median value of owner-occupied homes in $1000's\n\n    :Missing Attribute Values: None\n\n    :Creator: Harrison, D. and Rubinfeld, D.L.\n\nThis is a copy of UCI ML housing dataset.\nhttps://archive.ics.uci.edu/ml/machine-learning-databases/housing/\n\n\nThis dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.\n\nThe Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic\nprices and the demand for clean air', J. Environ. Economics & Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics\n...', Wiley, 1980.   N.B. Various transformations are used in the table on\npages 244-261 of the latter.\n\nThe Boston house-price data has been used in many machine learning papers that address regression\nproblems.   \n     \n.. topic:: References\n\n   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.\n   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.\n", 'filename': 'C:\\Users\\lenovo\\anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\boston_house_prices.csv'}
```


In [3]:
```python
#Loading the dataset into pandas dataframe
```

In [4]:
```python
hpp2d= pd.DataFrame(hp2.data,columns= hp2.feature_names)
```

In [5]:
```python
hpp2d.shape
```

Out[5]: (506, 13)

In [6]:
```python
hpp2d.head()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [7]:
```python
#Add target column to the dataframe
hpp2d['price']=hp2.target
```

In [8]:
```python
hpp2d.head()
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [9]:
```python
hpp2d.shape
```

Out[9]: (506, 14)

In [10]:
```python
hpp2d.isnull().sum().any()
```

Out[10]: False

In [11]:
```python
#Statistical measures of the datasets
hpp2d.describe()
```

Out[11]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

In [12]:
```python
#finding some info about data

hpp2d.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    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    float64
 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  price    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

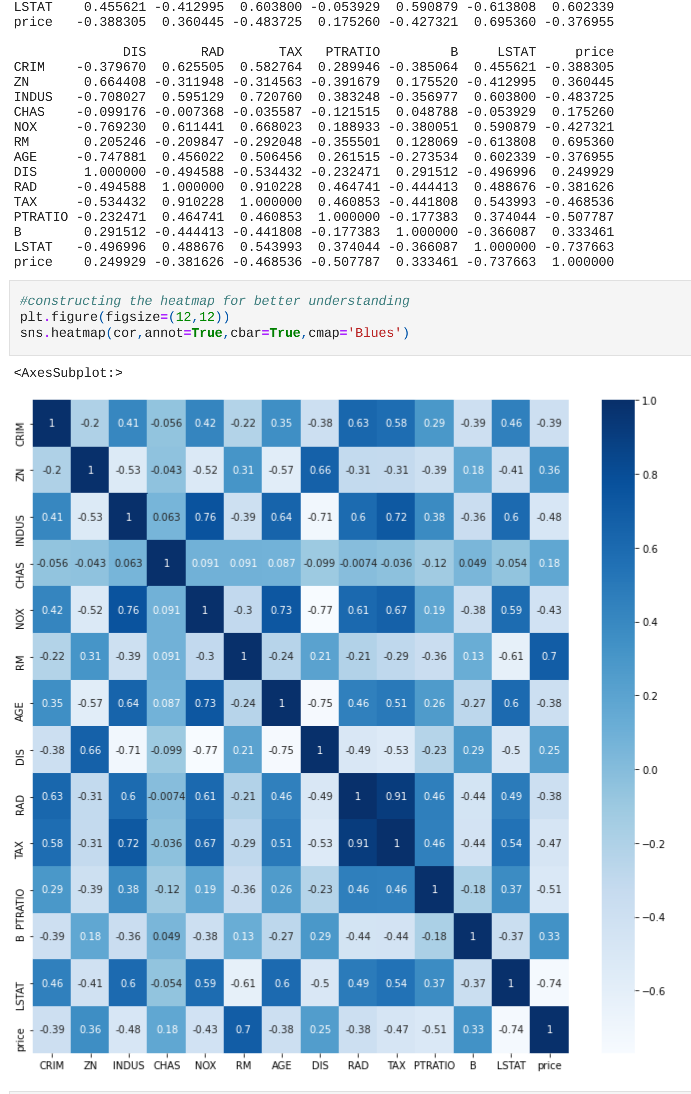
In [13]:
```python
#checkng correalaton of data
cor=hpp2d.corr()
print(cor)
```

```
             CRIM        ZN     INDUS      CHAS       NOX        RM       AGE  \
CRIM     1.000000 -0.200469  0.406583 -0.055892  0.420972 -0.219247  0.352734   
ZN      -0.200469  1.000000 -0.533828 -0.042697 -0.516604  0.311991 -0.569537   
INDUS    0.406583 -0.533828  1.000000  0.062938  0.763651 -0.391676  0.644779   
CHAS    -0.055892 -0.042697  0.062938  1.000000  0.091203  0.091251  0.086518   
NOX      0.420972 -0.516604  0.763651  0.091203  1.000000 -0.302188  0.731470   
RM      -0.219247  0.311991 -0.391676  0.091251 -0.302188  1.000000 -0.240265   
AGE      0.352734 -0.569537  0.644779  0.086518  0.731470 -0.240265  1.000000   
DIS     -0.379670  0.664408 -0.708027 -0.099176 -0.769230  0.205246 -0.747881   
RAD      0.625505 -0.311948  0.595129 -0.007368  0.611441 -0.209847  0.456022   
TAX      0.582764 -0.314563  0.720760 -0.035587  0.668023 -0.292048  0.506456   
PTRATIO  0.289946 -0.391679  0.383248 -0.121515  0.188933 -0.355501  0.261515   
B       -0.385064  0.175520 -0.356977  0.048788 -0.380051  0.128069 -0.273534   
LSTAT    0.455621 -0.412995  0.603800 -0.053929  0.590879 -0.613808  0.602339   
price   -0.388305  0.360445 -0.483725  0.175260 -0.427321  0.695360 -0.376955   

              DIS       RAD       TAX   PTRATIO         B     LSTAT     price  
CRIM    -0.379670  0.625505  0.582764  0.289946 -0.385064  0.455621 -0.388305  
ZN       0.664408 -0.311948 -0.314563 -0.391679  0.175520 -0.412995  0.360445  
INDUS   -0.708027  0.595129  0.720760  0.383248 -0.356977  0.603800 -0.483725  
CHAS    -0.099176 -0.007368 -0.035587 -0.121515  0.048788 -0.053929  0.175260  
NOX     -0.769230  0.611441  0.668023  0.188933 -0.380051  0.590879 -0.427321  
RM       0.205246 -0.209847 -0.292048 -0.355501  0.128069 -0.613808  0.695360  
AGE     -0.747881  0.456022  0.506456  0.261515 -0.273534  0.602339 -0.376955  
DIS      1.000000 -0.494588 -0.534432 -0.232471  0.291512 -0.496996  0.249929  
RAD     -0.494588  1.000000  0.910228  0.464741 -0.444413  0.488676 -0.381626  
TAX     -0.534432  0.910228  1.000000  0.460853 -0.441808  0.543993 -0.468536  
PTRATIO -0.232471  0.464741  0.460853  1.000000 -0.177383  0.374044 -0.507787  
B        0.291512 -0.444413 -0.441808 -0.177383  1.000000 -0.366087  0.333461  
LSTAT   -0.496996  0.488676  0.543993  0.374044 -0.366087  1.000000 -0.737663  
price    0.249929 -0.381626 -0.468536 -0.507787  0.333461 -0.737663  1.000000  
```

In [14]:
```python
#constructing the heatmap for better understanding
plt.figure(figsize=(12,12))
sns.heatmap(cor,annot=True,cbar=True,cmap='Blues')
```

Out[14]: <AxesSubplot:>



In [15]:
```python
from sklearn.model_selection import train_test_split
x=hpp2d.drop(['price'],axis=1)
y=hpp2d['price']
```

In [16]:
```python
print(x)
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0   
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0   
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0   
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0   
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0   
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ...   
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786  1.0  273.0   
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875  1.0  273.0   
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675  1.0  273.0   
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889  1.0  273.0   
505  0.04741   0.0  11.93   0.0  0.573  6.030  80.8  2.5050  1.0  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]
```

In [17]:
```python
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: price, Length: 506, dtype: float64
```

In [18]:
```python
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=2)
```

In [19]:
```python
#Model training
from xgboost import XGBRFRegressor
```

In [20]:
```python
#loading the model
xgb=XGBRFRegressor()
```

In [21]:
```python
xgb.fit(X_train,y_train)
```

Out[21]:
```
XGBRFRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bytree=1, gamma=0, gpu_id=-1, importance_type='gain',
               interaction_constraints='', max_delta_step=0, max_depth=6,
               min_child_weight=1, missing=nan, monotone_constraints='()',
               n_estimators=100, n_jobs=8, num_parallel_tree=100,
               objective='reg:squarederror', random_state=0, reg_alpha=0,
               scale_pos_weight=1, tree_method='exact', validate_parameters=1,
               verbosity=None)
```

In [22]:
```python
#evalution
xtrain=xgb.predict(X_train)
print(xtrain)
```

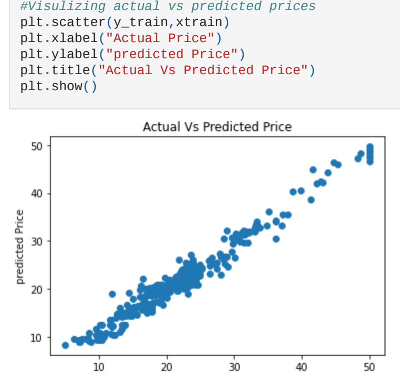
```
[21.359539  31.949036  16.358376  23.985476  20.87934   20.513628
 21.008251  35.500256  30.289135  32.37274   21.630157  49.01347
 14.28925    8.37607   22.245953   9.860197  25.320978  18.44369
 21.536407  45.953117  40.433773  34.010197  22.31657   33.408478
 24.17715   49.419594  14.064627  19.098562  20.838724  22.765743
 48.555817  23.2786    24.513334  20.59294   44.819138  19.897303
 31.440487  34.09962    8.965159  20.325293  18.135838  12.339044
 22.745676  47.739147  35.440758  20.427423  38.604023  19.930319
 15.675181  26.46901   20.804934  20.482819  20.004486  19.389997
 20.807354  21.11513   19.005182  32.832306  17.978191  13.557215
 19.336042  20.618116  17.03843   20.613392  49.46881   16.22802
 15.941126  20.273365  14.519324  31.94642   15.847171  44.22469
 33.324047  19.79377   15.596699   9.515258  10.718094  13.65734
 20.02701    9.536958  21.706581  11.131923  20.583231  47.4348
 21.676022  20.288391  31.680279  20.843147  31.858261  30.595032
 14.937328  16.4413    48.97171   21.217867  16.664581  11.606398
 49.39708   31.520895  10.51585   20.422909  29.507177  32.088272
 21.823551   9.325565  23.984375  23.789701  22.77846   12.447714
 26.778223  33.38871   15.343241  20.32534   26.05399   33.31414
 29.813885  22.965065  10.880626  24.063171  20.909239  34.29747
 23.842974  20.499273  20.316216   9.945973  21.100399  18.705935
 20.994663  11.677767  42.44933   48.249355  14.337326  20.809786
 23.882547  16.689676  20.822447  22.06787   32.879093  20.889698
 24.244593  18.774384  31.94337   15.053405  14.991616  21.810673
 25.415352  16.664993  25.89868   24.339472  26.653059  21.472063
 18.967703  26.51944   14.433957  19.656963  17.318403  42.15245
 24.68092   20.949781  25.097628  19.365797  19.941952  20.261227
 21.910706  24.927464  17.687996  14.996196  20.98887   21.649143
 10.9047785 18.977985  20.86647   20.800322  20.499264   9.686793
 23.00317   20.675829  16.163366  24.246872  30.541475  22.47135
 14.539515  19.333937  26.1184    20.833256  46.642105  20.625093
 31.509129  49.675377  16.364782  17.06529    9.306274  20.450657
 16.54414   15.975145  16.45014   20.719223  30.400991  26.744152
 19.783861  20.380949  23.785767  10.345345  25.695774  47.715683
 21.310078  23.663565  20.595682  12.522885  19.287668  15.046336
 14.754101  21.836653  15.626396  20.838627  25.994751  17.18033
 23.761559  15.021115  20.965496  20.344646  25.115036  27.417168
 22.822905  18.867483  23.832256  24.9051    16.07872   18.35211
 23.905325  15.696151  23.796906  29.579174  15.28884   25.361557
 16.229267  14.352384  10.906681  23.786533  14.598803  36.131226
 20.536888  41.91707    8.915807  20.986912  15.20648   14.875705
 16.842052  23.548777  22.413055  46.44488   19.077103  29.602041
 25.452595  20.101028  26.087448  10.46756   40.287495  20.101856
 18.442308  25.504972  19.093922  23.972528   9.064347  23.822748
 20.144272  20.9435    23.878283  21.799805  18.647684  18.572191
 15.083117  25.256178  14.608524  23.080055  20.957628  14.919534
 17.747595  23.423494  24.230808  22.276403  23.660397  30.671024
 20.646704  20.238401  27.88154   23.691572  12.538449  22.613531
 12.039306   9.405318  21.082556   9.45362   47.181843  24.482382
 12.559799  16.94203   14.984181  26.839815  19.395615  20.068163
  9.495608  20.720642  21.075245  20.3332    23.808512  24.4084
 26.837273  14.4747    19.296026  21.735142  18.490437  25.03376
 14.461924  23.110962  32.898174  27.118092  24.10951   20.250067
 20.442606  12.530785  23.746943  16.089817  20.942444  20.76243
 21.857718  18.562317  12.50955   32.778164  20.696022  48.49776
 20.794088  22.404352  21.246395  19.106588   9.123306  19.604073
 23.723051  21.215036  18.035936  22.904167  20.772095  26.349993
 14.875896  49.0149    31.054253  23.15839   22.616316  17.075817
 32.144157  16.353508  23.13484   20.761852  20.445658  21.772152 ]
```

In [23]:
```python
# R Squared Error
from sklearn import metrics
sc=metrics.r2_score(y_train,xtrain)
#MSE
sc1=metrics.mean_absolute_error(y_train,xtrain)
```

In [24]:
```python
print("The calculated R squared error are",sc)
print("The calculated MSE are",sc1)
```

```
The calculated R squared error are 0.9648485826144998
The calculated MSE are 1.3390544627345886
```

In [25]:
```python
#Visulizing actual vs predicted prices
plt.scatter(y_train,xtrain)
plt.xlabel("Actual Price")
plt.ylabel("predicted Price")
plt.title("Actual Vs Predicted Price")
plt.show()
```



In [26]:
```python
xtest=xgb.predict(X_test)
print(xtest)
```

```
[20.934166  20.765793  32.99709   28.58591    8.993672  14.177376
 27.516039  29.592365  29.249186  20.475286  25.450895  24.291237
 20.768852  20.918112  11.027329  22.784014  17.862402  10.792487
  9.449298  15.432918  24.063171  20.527916  33.5453    19.115477
 15.0263815 20.476936  41.64321   33.37461   33.442898  20.645563
 19.573997  20.651598  30.442425  24.580706  12.353749  16.638952
 10.235276  20.469995  21.328396  21.552038  24.595903  15.713804
 31.472038   9.1368475 21.178349  14.596579  34.601826  16.239532
 31.169796  15.070799  29.275543  27.726635   8.643508  32.567085
 31.5166    20.341454  20.374031  20.046034  17.196047  20.250797
 19.960463  22.163408  19.835064  28.982603  31.317839  24.694933
 47.819523  27.754173  10.706607  22.811312  16.296736   9.432871
 20.702284  18.19916   25.422634  25.572319  20.877977  21.140223
 20.76536   22.83164   33.552704  18.020735  21.866024  29.380175
 46.806908  35.86994   20.048134  22.839872  30.55053   20.572489
 20.718298  20.63918   10.66596   35.531578  37.335804   9.653868
 42.67209   33.972317  19.579287  16.267544  29.93828   23.94093
 23.924416  20.929237  23.293795  34.668533  16.516163  16.541462
 22.6106    14.60067   16.466362  21.243523  16.167091   9.363468
 27.43483   40.847897  31.94506   20.576561  20.799318  20.922298
 33.112392  20.441698  24.486229   9.61153   43.766487  20.963377
 17.463734  20.701868  21.04779   27.212091  21.080532  32.25708
 21.51796    9.982764  20.451326  18.035141  11.58059    9.939402
 12.784132  21.49      18.210255  14.976512  23.880587  34.7173
 16.538185  16.73099   16.37392   16.15171   26.786856  17.352398
 45.007927  18.96881  ]
```

In [27]:
```python
# R Squared Error
sc=metrics.r2_score(y_test,xtest)
#MSE
sc1=metrics.mean_absolute_error(y_test,xtest)
```

In [28]:
```python
print("The calculated R squared error are",sc)
print("The calculated MSE are",sc1)
```

```
The calculated R squared error are 0.8854102642208632
The calculated MSE are 2.258268358832911
```