In [21]:
```
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 xgboost import XGBRegressor
from sklearn import metrics
```

In [10]:
```
house_price_dataset=pd.read_csv("Boston Housing.csv")
```

In [11]:
```
print(house_price_dataset)
```

```
        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
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       ...     ...    ...   ...
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

In [15]:
```
house_price_dataset.rename(columns={"medv":"price"},inplace=True)
```

In [16]:
```
print(house_price_dataset)
```

```
        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
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  price
0       15.3  396.90   4.98   24.0
1       17.8  396.90   9.14   21.6
2       17.8  392.83   4.03   34.7
3       18.7  394.63   2.94   33.4
4       18.7  396.90   5.33   36.2
..       ...     ...    ...    ...
501     21.0  391.99   9.67   22.4
502     21.0  396.90   9.08   20.6
503     21.0  396.90   5.64   23.9
504     21.0  393.45   6.48   22.0
505     21.0  396.90   7.88   11.9

[506 rows x 14 columns]
```

In [17]:
```
house_price_dataset.shape
```

Out[17]: (506, 14)

In [18]:
```
house_price_dataset.describe()
```

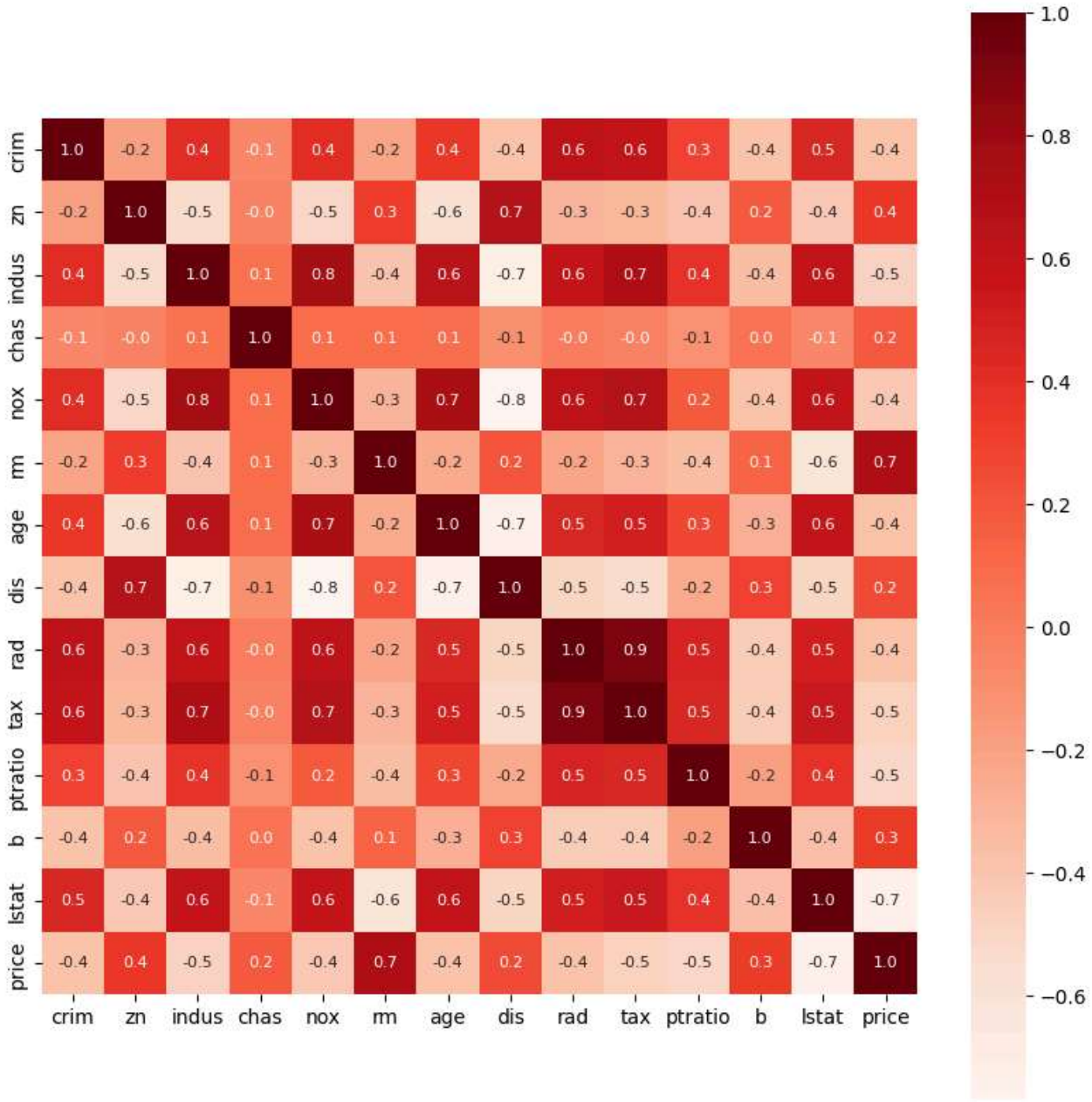
Out[18]:

m	zn	indus	chas	nox	rm	age	dis	rad	tax	ptrati
00	506.000000	506.000000	506.000000	506.000000	501.000000	506.000000	506.000000	506.000000	506.000000	506.00000
24	11.363636	11.136779	0.069170	0.554695	6.284341	68.574901	3.795043	9.549407	408.237154	18.45553
45	23.322453	6.860353	0.253994	0.115878	0.705587	28.148861	2.105710	8.707259	168.537116	2.16494
20	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.60000
45	0.000000	5.190000	0.000000	0.449000	5.884000	45.025000	2.100175	4.000000	279.000000	17.40000
10	0.000000	9.690000	0.000000	0.538000	6.208000	77.500000	3.207450	5.000000	330.000000	19.05000
33	12.500000	18.100000	0.000000	0.624000	6.625000	94.075000	5.188425	24.000000	666.000000	20.20000
00	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.00000

In [19]:
```
correlation=house_price_dataset.corr()
```

In [25]:
```
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,square=True,fmt=".1f",annot=True,annot_kws={"size":8},cmap="Red
```

Out[25]: <Axes: >



In [28]:
```
x=house_price_dataset.drop(["price"],axis=1)
y=house_price_dataset["price"]
```

In [29]:
```
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
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     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: price, Length: 506, dtype: float64
```

In [31]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

In [32]:
```
model=XGBRegressor()
```

In [33]:
```
model.fit(x_train,y_train)
```

Out[33]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=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,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

In [34]:
```
training_data_prediction=model.predict(x_train)
```

In [35]:
```
print(training_data_prediction)
```

```
[23.098213  20.999493  20.100712  34.69146   13.908512  13.499188
 22.005543  15.206342  10.8988695 22.677683  13.802292   5.5992246
 29.79903   50.000896  34.881298  20.619646  23.372395  19.202538
 32.697285  19.604185  26.987984   8.400937  45.993626  21.712433
 27.08105   19.349197  19.29314   24.7933    22.61487   31.693779
 18.539114   8.702092  17.40409   23.699408  13.297645  10.494993
 12.669688  25.000593  19.698307  14.9110775 24.21232   25.000761
 14.900763  17.013046  15.597457  12.68823   24.503881  14.999328
 49.99955   17.532228  21.199259  32.010696  15.601823  22.893911
 19.322012  18.73874   23.292349  37.210262  30.105167  33.121197
 20.999691  49.985893  13.400765   5.009729  16.502712   8.40547
 28.691751  19.494135  20.596006  45.400337  39.80286   33.40684
 19.829916  33.40482   25.281872  49.998436  12.519731  17.429428
 18.605316  22.591347  50.00362   23.805405  23.30907   23.092073
 41.70726   16.111097  31.623669  36.082043   6.9961967 20.400877
 19.989061  11.98513   25.046114  49.968384  37.89933   23.098452
 41.309887  17.590723  16.296797  30.049692  22.863256  19.800455
 17.10464   18.901573  18.911537  22.599606  23.177149  33.20601
 14.998805  11.705264  18.802998  20.812239  17.996387  19.630117
 49.998535  17.201649  16.404898  17.50357   14.599329  33.103542
 14.500418  43.812004  34.903797  20.394728  14.62216    8.093705
 11.792478  11.815979  18.68806    6.304934  23.985     13.07154
 19.601086  49.98379   22.312391  18.926582  31.205538  20.703587
 32.201614  36.17072   14.194604  15.699518  49.991993  20.413244
 16.1949    13.396754  49.992847  31.607615  12.266604  19.208345
 29.793688  31.506674  22.800661  10.199806  24.09686   23.701332
 22.00898   13.798573  28.402477  33.16191   13.103114  19.000732
 26.61407   36.977833  30.79909   22.790192  10.204003  22.203903
 24.473537  36.195976  23.099493  20.124691  19.486431  10.798922
 22.708363  19.520206  20.105482   9.618499  42.799683  48.787125
 13.107633  20.297323  24.777777  14.100037  21.696146  22.159771
 33.000343  21.096354  24.987467  19.10297   32.395473  13.60095
 15.076382  23.065681  27.493874  19.400146  26.486158  27.50132
 28.696423  21.219297  18.703724  26.69608   14.009387  21.687016
 18.36929   43.097454  29.084362  20.295773  23.70212   18.305271
 17.203133  18.31718   24.379559  26.407068  19.093657  13.302053
 22.151814  22.193771   8.535822  18.900906  21.794275  19.301249
 18.198235   7.5031695 22.39194   20.005772  14.405179  22.504189
 28.496006  21.609188  13.797945  20.4931    21.902758  23.101397
 49.996716  16.235315  30.304276  50.010593  17.784084  19.066183
 10.402657  20.361696  16.504766  17.198158  16.73516   19.542725
 30.511305  29.022295  19.556704  23.16179   24.410498   9.503701
 23.889511  49.992424  21.190037  22.59402   20.004858  13.400503
 19.992882  17.11022   12.705852  23.011364  15.190408  20.60995
 26.204994  18.10329   24.090755  14.101646  21.70381   20.07812
 24.99523   27.912807  22.92296   18.506147  22.195614  23.998285
 14.794136  19.883074  24.399878  17.789375  24.592009  32.00416
 17.786402  23.310347  16.102678  13.006888  11.000934  24.287457
 15.598231  35.21383   19.607826  42.306316   8.799025  24.411901
 14.106117  15.4000225 17.29131   22.142748  23.090525  44.821854
 17.797606  31.496786  22.788765  16.812382  23.909449  12.09243
 38.696747  21.405613  15.998646  23.922968  11.903084  24.975605
  7.2051916 24.70098   18.194902  22.4938    23.03588   24.301077
 17.096863  17.798708  13.505205  27.100895  13.301572  21.899876
 20.003866  15.368651  16.601213  22.29691   24.686428  21.391602
 22.885778  29.599918  21.8974    19.871151  29.603804  23.415052
 13.800514  24.492027  11.905178   7.2101088 20.497765   9.713653
 48.292183  25.205778  11.696702  17.40583   14.498054  28.578287
 19.38771   22.479158   7.028048  20.606165  22.99739   19.711414
 23.702559  25.042511  27.972979  13.398359  14.5203905 20.295069
 19.307014  24.094303  14.902324  26.38988   33.303482  23.635433
 24.600914  18.504745  20.897694  10.403686  23.30222   13.105769
 24.69939   22.596542  20.494177  16.816057  10.196415  33.827606
 18.601995  49.997982  23.782066  23.89807   21.177568  18.80857
  8.506363  21.496952  23.213255  21.040653  16.612797  28.085756
 21.204151  28.38265   14.288944  50.006256  30.999334  25.007833
```

```
21.433289  19.00253   28.993366  15.21607   22.79087   21.794004
19.900063  23.777464 ]
```

In [36]:
```
score_1=metrics.r2_score(y_train,training_data_prediction)
score_2=metrics.mean_absolute_error(y_train,training_data_prediction)
print("R squared error",score_1)
print("mean absolute error",score_2)
```

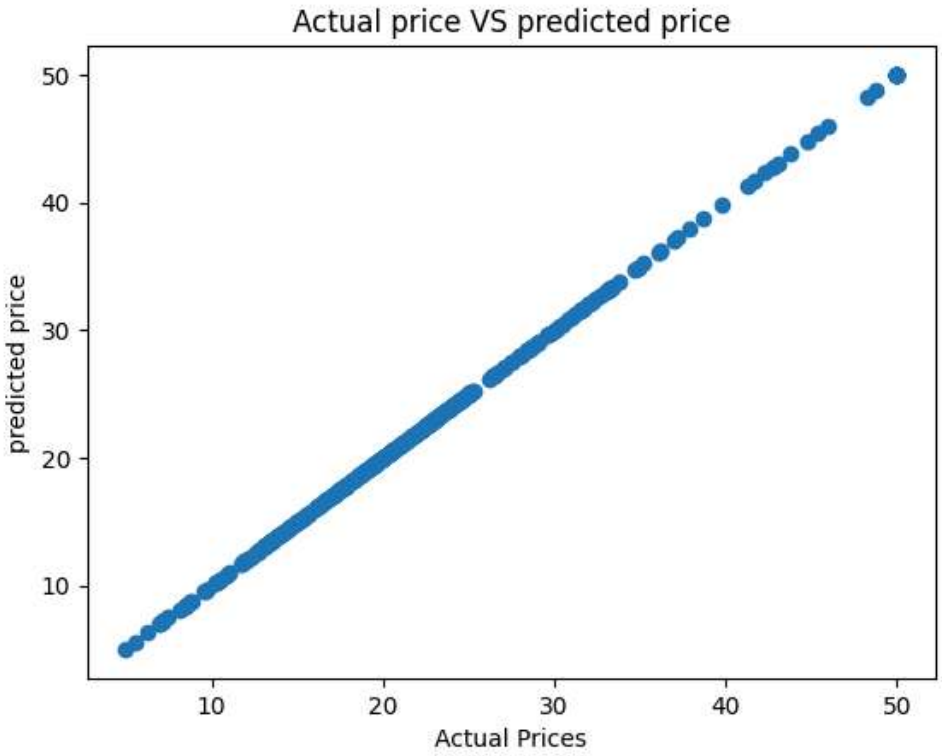
```
R squared error 0.9999974930039426
mean absolute error 0.0099853890957219
```

In [37]:
```
test_data_prediction=model.predict(x_test)
```

In [38]:
```
score_1=metrics.r2_score(y_test,test_data_prediction)
score_2=metrics.mean_absolute_error(y_test,test_data_prediction)
print("R squared error",score_1)
print("mean absolute error",score_2)
```

```
R squared error 0.9156404620793328
mean absolute error 1.9900942007700595
```

In [39]:
```
plt.scatter(y_train,training_data_prediction)
plt.xlabel("Actual Prices")
plt.ylabel("predicted price")
plt.title("Actual price VS predicted price")
plt.show()
```



In [ ]:

In [ ]: