In [42]:
```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn.metrics import accuracy_score
from sklearn import metrics
```

In [35]:
```python
df=pd.read_excel("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\Excel file\\Boosten house price.xlsx")
```

In [3]:
```python
df.head()
```

Out[3]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	price
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	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	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	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	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	18.7	396.90	5.33	36.2

In [6]:
```python
df.shape
```

Out[6]:
```
(506, 14)
```

In [7]:
```python
#check the missing value
df.isnull().sum()
```

Out[7]:
```
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
price      0
dtype: int64
```

In [9]:
```python
#describe mathamatical data
df.describe()
```

Out[9]:

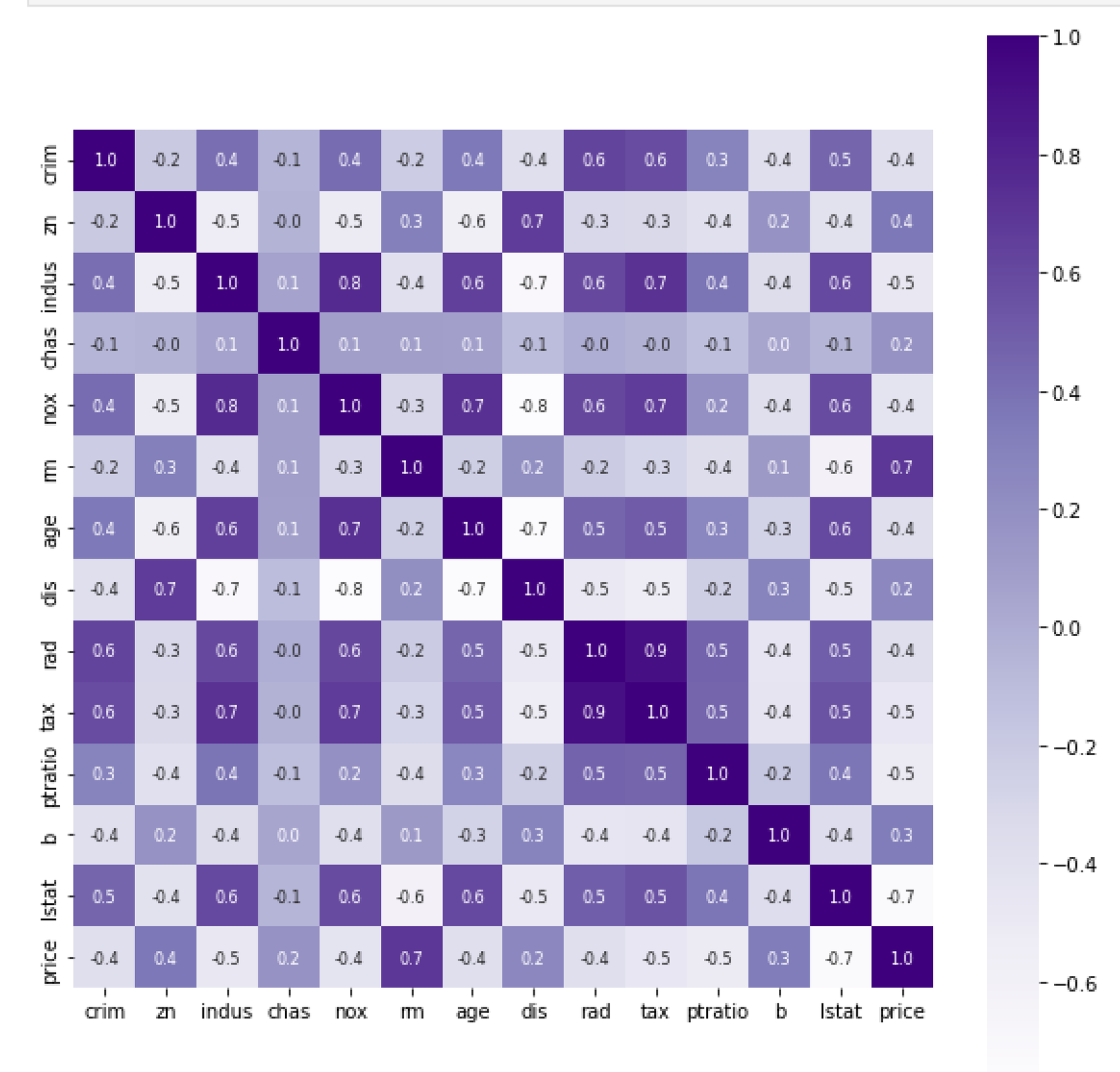
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	price
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	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 [11]:
```python
#Understanding the correlation between various features in the data set 1-->Positive correlation  2-->Negative correlation
correlation=df.corr()
print(correlation)
```

```
             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 [17]:
```python
#heatmap
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Purples')
plt.show()
```



In [21]:
```python
#splitting the data
X=df.drop(['price'],axis=1)
y=df['price']
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 [26]:
```python
#training and test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (404, 13)
shape of X_test=  (102, 13)
shape of y_train=  (404,)
shape of y_test=  (102,)
```

In [30]:
```python
#model training  XGBoost Regressor
model=XGBRegressor()
```

In [32]:
```python
model.fit(X_train,y_train)
```

Out[32]:
```
XGBRegressor
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

In [37]:
```python
#Predction on training data
X_train_prediction=model.predict(X_train)
print(X_train_prediction)
```

```
[25.245571  23.32921    7.210682  21.154896  11.702352  26.999006
 29.584614  26.544731  43.50524   23.625568  11.002917  33.40945
 36.007214  36.40028   19.037712  20.18499   34.905746  49.99532
 19.321676  14.898901  26.617535  19.894657  24.802687  21.180086
 23.923895  20.586973  23.074522  27.950964  19.987543  23.094906
 25.010534   9.705043  23.892748  36.09172   13.418543  12.731809
 39.79168   10.393237  20.598629  17.775103  19.482315  23.710855
 28.513987  24.29506   23.771362  19.072584  28.377209  20.49642
 33.825535  14.507262  20.39619   16.009405  13.304443  30.793392
 27.49651   24.428276  24.389343  25.122246  43.8164    21.898586
 26.22088   14.255268  20.789656  20.105751  23.099632  13.068087
 16.195873  24.834087  20.208426  22.484184  14.791592  28.696594
 20.114979  23.412987  31.992868  19.124737  49.992138  20.895212
 21.69236   21.999992  17.206669  30.303902  12.287807  21.39682
 20.501791  35.200222  19.602581  22.003605  21.689674  14.086667
 21.088537  15.00701   11.9128065 19.984388  41.284725  18.702425
 50.006096  49.998714  18.385103  17.894514  28.100218  16.111639
 17.235275  28.587082  23.623482  20.418549  19.594658  18.8178
 22.577982  17.710707  30.53399   18.24904   20.639639  24.39996
 17.3008    13.326973  22.788692  20.502954  21.194191  18.78094
 18.910189  18.230228  23.098423  32.71051   23.99935   10.215775
 19.499582  33.106304  13.388557  15.1955385 24.802052  24.24765
  9.519843  24.201674  18.508362  43.98555   49.996807  24.680128
 21.516027   8.403035  21.78793   49.998203  23.771406  32.39565
 24.405802  17.586105  29.789497   9.629779  16.695997  13.8042
 32.004784  16.108585   8.289033  26.604982  14.297345  15.001556
 28.400133  32.204517  17.095285  29.387379  10.39832   16.81697
 31.507769  27.471685  46.689636  27.428337  17.180197  23.359629
 31.617931  13.802507  21.99958   16.983261  24.799078  24.29832
 25.20803   21.238384  20.61462   18.711704   5.613344  19.308567
 19.816687  22.295553  20.287615  12.028596  23.885447  16.49773
 13.206723  33.197372  10.501575   7.5048037 27.450851  18.397408
 23.21327   13.791749  35.40658   22.976023  25.000204   7.2061105
 14.401987   8.780611  22.698383  13.090879  18.963488  24.983158
  8.501394  16.099056  28.98103   23.096695  19.308617  33.099937
 24.582895  23.00888   15.210551  27.089996  19.552542  24.460632
 20.26648   34.888847  17.10871   15.600463  26.397736  22.621044
 15.595059  29.001913  21.227331  22.416351  13.495529  11.671535
 17.099998  31.736755  28.654648  24.696526  18.99535    7.224414
 13.805055  12.783236  36.168663  38.681366  18.536562  29.062824
 20.374943  11.307514  17.43838    8.700741  18.905281  23.208126
 22.137459  29.109512  34.58174   24.992807  23.187595  37.892902
  7.004926  18.195997  19.325453  26.717451  19.198877  30.102848
 20.602144  49.99454   18.683456  20.591879  31.083271  14.002027
 17.764019  42.294308  15.306528  18.501596  21.354275  14.997089
 20.685452  21.397749  21.708717  22.01213   31.598022  22.024744
 10.191377  22.605482  20.009838  17.79616   13.597596  11.773635
 19.39328   21.427917  32.920986  20.793856  30.985199  17.525541
 15.406584  10.804339  34.683525  25.026157  48.81037   42.79827
 19.50153   30.110062  22.198988  50.006954  23.106049  32.49651
 19.600311  14.910648  26.389574  36.957565  24.09423   24.493849
 23.708567   6.9975147 22.253477  23.338387  15.58326   13.387992
 30.706673  22.284441  17.403036  50.01416   22.897459  19.690071
 15.602127  17.805304  10.900939  35.10736   15.701772  49.989758
 22.802149  19.903727  20.10049   19.377815  46.00134   23.145538
 37.58125   23.115402  13.892105  33.323578  32.995598  19.906973
 20.316751  49.98246   19.393984  19.480566  22.80591   16.602104
 19.999386  24.694563  45.401737  33.398872  21.387144  19.392666
  5.0454464  7.4002113 20.085537  12.727631  20.287464  14.121794
 18.30175   19.887678  23.305897  36.46754   20.000673  17.807898
  8.794243  21.6059    21.593664  15.202687  19.801857  21.044683
 27.086187  16.858004  14.393107  22.468117  18.600527  20.073027
 19.649809  25.012434  17.433014  19.717339   4.987939  16.30734
 13.10698   29.595028  13.075437  19.095324  12.078261  21.690704
 21.90341   33.199932  29.90732   35.391438  15.074145  31.495838
 21.693008  16.398607  14.334121  11.802701  14.101409  21.105223
 18.343138  48.511383  13.876459  20.894222  22.81594   12.522005
 24.028767  20.946033 ]
```

In [45]:
```python
#R Squared error
score_1=metrics.r2_score(y_train,X_train_prediction)
#mean Absolute error
score_2=metrics.mean_absolute_error(y_train,X_train_prediction)
print("R Squared error ",score_1)
print("mean Absolute error",score_2)
```

```
R Squared error  0.9999952977725205
mean Absolute error 0.01375658535721277
```

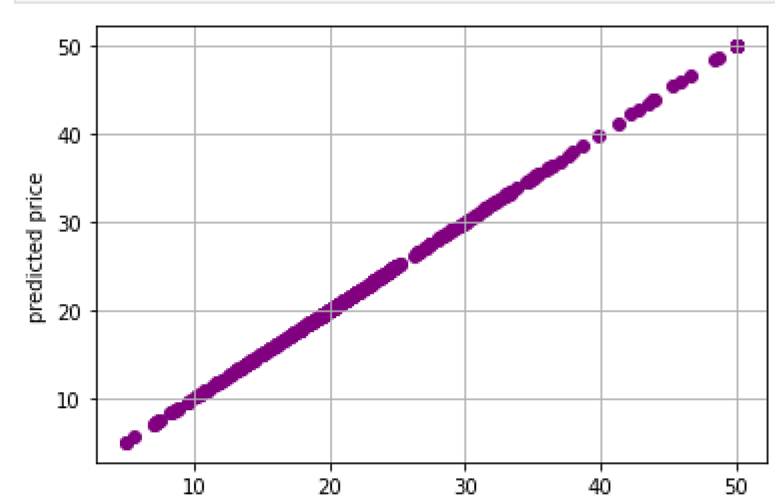
In [46]:
```python
#Predction on testing  data
X_test_prediction=model.predict(X_test)
print(X_test_prediction)
```

```
[30.359884 27.026636 19.17616  21.448114 20.703493 20.424747 27.9036
 17.878405 21.678137 23.36513  25.411674 33.19199  21.495481 19.939993
 19.990822 28.529932 13.219593 44.483154 25.460302 11.562899 19.486588
 17.153116 24.325785 23.509218 27.791344  8.616411 14.683976 19.01615
 45.651524 12.10203  22.976582 17.523937 48.34446  16.573458 24.024517
 21.818398 14.679352 35.878193 16.073364 20.764208 25.262817 22.376215
 24.78839  14.799538 16.877083 11.262062 46.995094 11.280138 21.217411
 18.637756 23.77302  20.691387 25.980684 20.501291  8.173699 22.731295
 11.759897 24.526464 18.271172 38.32712  12.570583 25.870787 12.011916
 13.295293 17.679295 35.901722 37.55958  23.164257 20.201231 20.955679
 24.68425   7.049963 18.19561  19.824377 19.77468  20.867382 40.98662
 24.473787 27.369884 31.096327 16.441713 19.326612 36.752098  9.891363
 20.98247  25.28127  14.024525 26.155094 21.51227  16.4159   23.28247
 45.429245 15.133404 20.947332 14.996857 20.791437 24.775454 23.958286
 45.176277 22.56781  15.780695 16.750847]
```

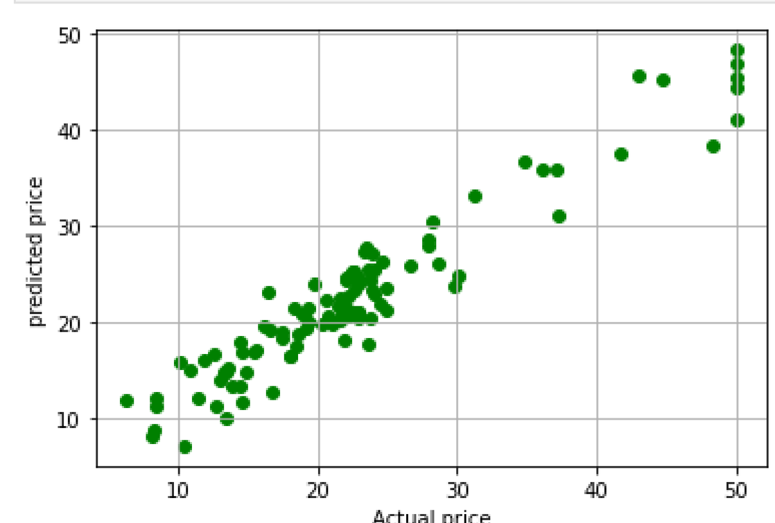
In [47]:
```python
#R Squared error
score_3=metrics.r2_score( y_test,X_test_prediction)
#mean Absolute error
score_4=metrics.mean_absolute_error( y_test,X_test_prediction)
print("R Squared error ",score_3)
print("mean Absolute error",score_4)
```

```
R Squared error  0.9085331557145866
mean Absolute error 2.344013180452235
```

In [52]:
```python
plt.scatter(y_train,X_train_prediction,color='purple')
plt.xlabel('Actual price')
plt.ylabel('predicted price')
plt.grid(True)
plt.show()
```



In [53]:
```python
plt.scatter( y_test,X_test_prediction,color='green')
plt.xlabel('Actual price')
plt.ylabel('predicted price')
plt.grid(True)
plt.show()
```



In [ ]: