In [2]:

```
import warnings
warnings.filterwarnings("ignore")
from sklearn.datasets import load_boston
from random import seed
from random import randrange
from csv import reader
from math import sqrt
from sklearn import preprocessing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
from sklearn.linear_model import SGDRegressor
from sklearn import preprocessing
from sklearn.metrics import mean_squared_error
```

In [3]:

```
X = load_boston().data
Y = load_boston().target
```

In [45]:

```
from sklearn.model_selection  import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X, Y, test_size=0.4, random_state=23)
```

In [46]:

```
scaler = preprocessing.StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test= scaler.transform(X_test)
df_train=pd.DataFrame(X_train)
df_train['price']=y_train
df_train.head()
```

Out[46]:

	0	1	2	3	4	5	6	7	8	9	10	11	
0	-0.408368	-0.511177	-0.867322	-0.30637	-0.322124	-0.376122	-1.158271	0.975013	-0.514591	-1.090186	0.797044	0.431450	-0.6
1	-0.293865	-0.511177	-0.438618	-0.30637	-0.128143	-0.528845	-1.342010	0.273016	-0.629833	-0.595391	1.167024	0.326727	-0.8
2	0.086377	-0.511177	1.008804	-0.30637	0.259819	-0.427030	0.961797	-0.651664	1.674989	1.544746	0.797044	-0.252221	1.2
3	-0.352545	0.319537	-1.044617	-0.30637	0.791158	1.844337	0.781591	-0.840774	-0.514591	-0.851730	-2.532774	0.342565	-0.7
4	-0.330906	-0.511177	-0.438618	-0.30637	-0.128143	-0.840278	0.078436	-0.058505	-0.629833	-0.595391	1.167024	0.369450	-0.1

In [53]:

```
# Reference:
https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
clf = SGDRegressor(n_iter=500)
clf.fit(X_train, y_train)
y_pred_SGD= clf.predict(X_test)
MSE_SGD=mean_squared_error(y_test,y_pred_SGD)
```

```
C:\Program Files\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:117:
DeprecationWarning: n_iter parameter is deprecated in 0.19 and will be removed in 0.21. Use
max_iter and tol instead.
  DeprecationWarning)
  DeprecationWarning)
```

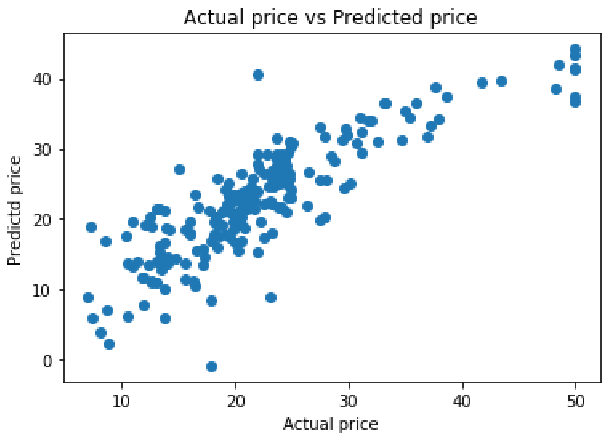
In [54]:

```
print(MSE_SGD)
```

```
22.29025363240749
```

In [55]:

```
plt.scatter(y_test,y_pred_SGD)
plt.xlabel('Actual price')
plt.ylabel('Predictd price')
plt.title('Actual price vs Predicted price')
plt.show()
```



In [72]:

```
#Reference:https://towardsdatascience.com/step-by-step-tutorial-on-linear-regression-with-stochast
ic-gradient-descent-1d35b088a843
W,B, Learn_rate,Num_iterations,k=np.zeros(shape=(1,13)),0,0.01,500,30
weight=[]
while Num_iterations>=0:
    w, b , diff_dw , diff_db = W,B, np.zeros(shape=(1,13)) ,0
    data_train=df_train.sample(30)
    x=np.array(data_train.drop('price',axis=1))
    y=np.array(data_train['price'])
    for i in range(k):
        diff_dw += (-2) * x[i]*(y[i]- np.dot(w,x[i])-b)
        diff_db += (-2) *(y[i]- np.dot(w,x[i])-b)

    W= (w - Learn_rate * (diff_dw/k))
    B= (b - Learn_rate * (diff_db/k))
    weight.append(W/13)
    Num_iterations=Num_iterations-1
```

In [73]:

```
#Reference : https://docs.scipy.org/doc/numpy/reference/generated/numpy.dot.html # y is array of s
ize 1 thus converting into scalar value using asscalar().

# y is array of size 1 thus converting into scalar value using asscalar().
https://www.geeksforgeeks.org/numpy-asscalar-in-python/
Y_pred=[]
for i in range(len(X_test)):
    Y_obtained= np.dot(W,X_test[i]) + B
    Y_pred.append(np.asscalar(Y_obtained))
```

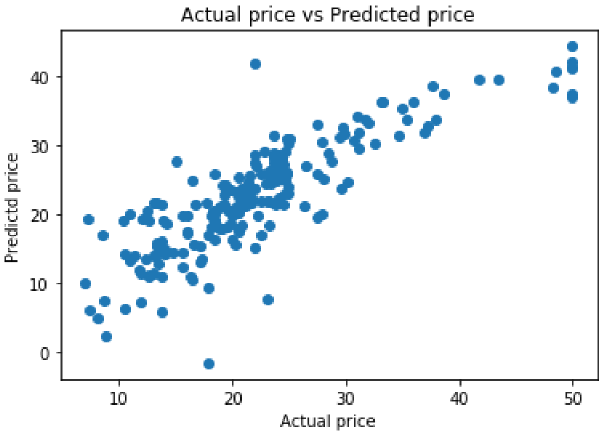
In [74]:

```
print(mean_squared_error(y_test,Y_pred))
```

```
23.213191854841085
```

In [75]:

```
#Scatter plot for actual and predicted
plt.scatter(y_test,Y_pred)
plt.xlabel('Actual price')
plt.ylabel('Predictd price')
plt.title('Actual price vs Predicted price')
plt.show()
```



In [77]:

```
#coef_ is to find weights assigned in classifier, obtained from documentation of SGD classifier
from prettytable import PrettyTable
x = PrettyTable()
x.field_names=['Manually calculated Weight Vector','SGD sklearn Weight Vector']
weight_sgd=clf.coef_
for i in range(13):
    x.add_row([W[0][i],weight_sgd[i]])
print(x)
```

```
+-----------------------------------+--------------------------+
| Manually calculated Weight Vector | SGD sklearn Weight Vector |
+-----------------------------------+--------------------------+
|         -0.615687110007786        |    -0.769664687430443    |
|         0.6012476406353022        |    0.8531202073742427    |
|        -0.36042125897752114       |   -0.0810799082800221    |
|         1.0183107601406394        |    0.8933151504072859    |
|         -1.469502607131612        |    -2.066780825279437    |
|         3.1378147740564373        |    2.941915713555898     |
|        -0.22022605155370684       |   -0.08655484818907985   |
|         -2.64178827680131         |   -2.9955901981520383    |
|         1.4785769031214984        |     2.2442444544317      |
|        -0.34725667555332024       |   -1.1430660443801108    |
|        -1.7743913940414004        |    -1.871916520301887    |
|         1.6016332435324914        |    1.5249663961736148    |
|         -3.816663401632293        |    -3.878864262298584    |
+-----------------------------------+--------------------------+
```