In [1]:
```
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
from sklearn.model_selection import train_test_split
```

In [2]:
```
# loading data into X,Y
X = load_boston().data
Y = load_boston().target
```

In [3]:
```
#splitting data into X_train and X_test
import sklearn
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.33, ra
ndom_state = 5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(339, 13)
(167, 13)
(339,)
(167,)
```

In [8]:
```
#standardizing the data
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

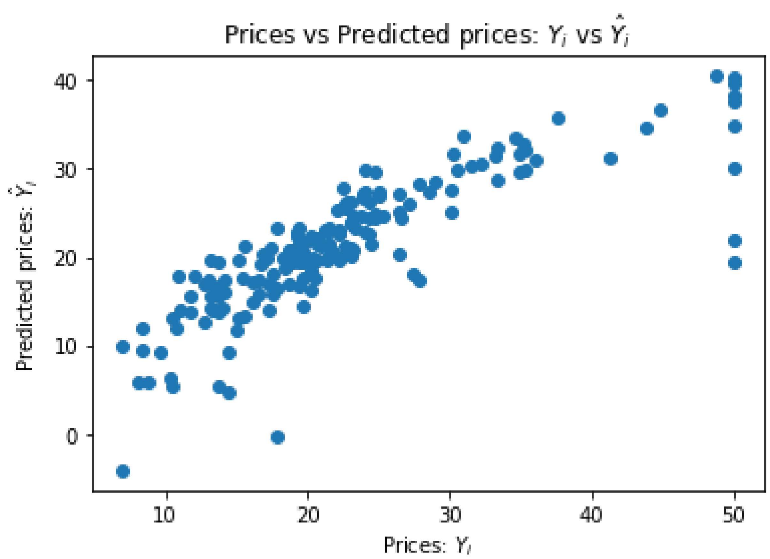
In [9]:
```
#clf = SGDRegressor(loss='squared_loss')
#clf.fit(X, Y)
#print(mean_squared_error(Y, clf.predict(X)))
```

In [12]:
```
#Applying SGD regressor on Train data
clf = SGDRegressor(loss='squared_loss')
clf.fit(X_train, Y_train)
```

Out[12]:
```
SGDRegressor(alpha=0.0001, average=False, epsilon=0.1, eta0=0.01,
       fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling',
       loss='squared_loss', max_iter=None, n_iter=None, penalty='l2',
       power_t=0.25, random_state=None, shuffle=True, tol=None, verbose=0,
       warm_start=False)
```

In [13]:
```
clf.fit(X_train, Y_train)
#Finding the predicted prices
Y_pred_clf = clf.predict(X_test)

#plotting a chart of actual prices vs predicted prices for SGD regressor
plt.scatter(Y_test, Y_pred_clf)
plt.xlabel("Prices: $Y_i$")
plt.ylabel("Predicted prices: $\hat{Y}_i$")
plt.title("Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$")
plt.show()
```

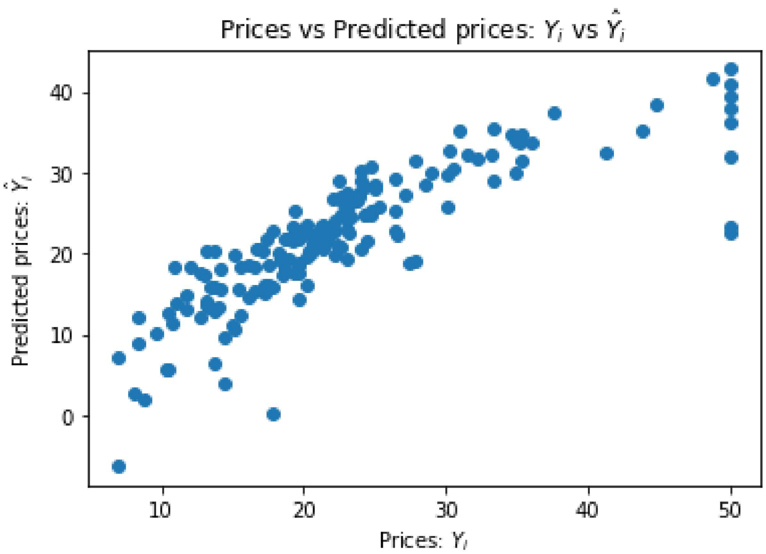


In [14]:
```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
lm.fit(X_train, Y_train)

#Finding the predicted price
Y_pred_lm = lm.predict(X_test)

#plotting a chart of actual prices vs predicted prices for linear regression
plt.scatter(Y_test, Y_pred_lm)
plt.xlabel("Prices: $Y_i$")
plt.ylabel("Predicted prices: $\hat{Y}_i$")
plt.title("Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$")
plt.show()
```



In [15]:
```
#Finding weight vector of SGD
clf.coef_
```

Out[15]:
```
array([-0.81363518,  0.40405972, -0.51748626,  0.29339603, -0.22368237,
        3.14507661, -0.42221585, -1.68006996,  0.74576312, -0.52451597,
       -1.77986047,  0.87611157, -2.98815837])
```

In [16]:
```
#Finding weight vector of linear regression
lm.coef_
```

Out[16]:
```
array([-1.31193031,  0.86187745, -0.16719287,  0.18957843, -1.48658584,
        2.79131565, -0.32737703, -2.77204093,  2.97567549, -2.2727549 ,
       -2.13375869,  1.05842993, -3.33495407])
```

In [17]:
```
#https://stackoverflow.com/questions/41140647/python-printing-lists-with-tabul
ate


#Tabulating and comparing SGD and Linear regression weights
from tabulate import tabulate
headers = ['SGD Weights','Linear regression weigths']


table = zip(clf.coef_,lm.coef_)
print(tabulate(table, headers=headers, floatfmt=".4f"))
```

```
  SGD Weights    Linear regression weigths
-------------  ---------------------------
      -0.8136                      -1.3119
       0.4041                       0.8619
      -0.5175                      -0.1672
       0.2934                       0.1896
      -0.2237                      -1.4866
       3.1451                       2.7913
      -0.4222                      -0.3274
      -1.6801                      -2.7720
       0.7458                       2.9757
      -0.5245                      -2.2728
      -1.7799                      -2.1338
       0.8761                       1.0584
      -2.9882                      -3.3350
```

In [18]:
```
clf.fit(X_train, Y_train)
print(mean_squared_error(Y_train, clf.predict(X_train)))
```

```
21.39072011815044
```

In [19]:
```
lm.fit(X_train, Y_train)
print(mean_squared_error(Y_train, lm.predict(X_train)))
```

```
19.54675847353467
```

In [20]:
```
#calculating mean squared errors of both models
a=(Y_test-clf.predict(X_test))
b=(Y_test-lm.predict(X_test))

a=a**2
b=b**2
```

In [21]:
```
  #comparing MSE of SGD regressor and linear regression
from tabulate import tabulate
headers = ['SGD MSE','Linear regression MSE']


table = zip(a,b)
print(tabulate(table, headers=headers, floatfmt=".4f"))
```

```
  SGD MSE    Linear regression MSE
---------  -----------------------
   0.6603                   0.0176
   1.0227                  12.1909
  16.9378                  20.4322
 100.6309                  53.7519
   3.8395                   2.4659
  38.3353                  22.3665
   8.0225                   9.8563
   0.0056                   0.8594
  13.7791                  12.6495
   8.6762                   0.5588
   0.6498                   0.9899
   4.7137                   7.7641
   5.0103                   9.6807
  19.3146                   6.2054
   5.5706                  15.1025
   0.6116                   4.2494
 410.3948                 309.7293
   0.0879                   0.7770
  30.9380                  27.3459
  28.5613                  15.1458
  13.6688                  29.2147
  13.4311                  14.3116
  50.4955                  40.8117
   0.9498                   1.6113
   0.0909                   0.3423
   0.2017                   0.3870
   2.1558                   3.3439
   3.9715                   4.4596
   1.3825                   0.3322
   0.1089                  11.8969
  17.7918                   1.5997
   3.1109                   0.0064
   9.4612                  24.5529
  21.3465                  25.1473
   3.1443                   7.9492
   0.7905                   0.9731
   6.6289                  10.7388
   0.5360                   0.2552
  48.8440                  51.1024
   5.0086                   0.7585
  20.2667                  40.7924
 123.2739                 109.0770
   3.4386                   0.1671
   2.4091                   0.2696
   0.0134                   0.0088
  42.5192                  34.0705
 195.8100                 175.5499
  12.8017                   0.3066
   1.3089                   0.0026
   0.5680                   0.4035
   0.5155                   0.0001
   4.4839                   5.4124
   3.8192                   4.7375
   0.0911                   0.0097
   0.5679                   3.0117
```

2.5984	0.0055
65.8017	71.9992
0.3204	1.9132
0.0673	0.0085
3.8915	4.1037
12.0519	14.2992
4.4805	8.3210
4.9402	2.8286
94.9936	79.6567
7.1728	3.5752
4.5771	15.1269
3.2892	1.1141
0.1442	0.2851
0.2992	0.0053
2.9018	0.4932
3.2614	15.0716
11.8745	21.5886
3.5886	3.4967
115.1919	77.1025
136.8576	78.6721
8.7710	18.7876
13.3833	20.0324
1.6009	0.0954
12.8608	17.0084
4.5382	14.7832
0.0359	8.0436
104.5661	112.3588
0.6322	0.3682
5.4019	10.7180
23.8102	34.7275
10.2004	18.6629
0.1739	1.1758
3.4962	0.1004
0.8049	0.4569
3.9801	1.8368
203.0105	188.7152
3.6719	4.1783
6.6290	0.0850
0.4778	0.4683
15.3659	4.5165
9.8774	9.5345
11.0441	11.1907
0.5707	17.5903
6.6314	1.9465
1.7963	0.1687
5.4615	0.0084
4.1824	9.0657
1.0552	1.1016
67.8689	52.4643
35.3341	46.0920
0.4646	1.9360
21.1132	38.7513
989.9258	751.1242
20.6970	23.3340
0.4032	5.7133
16.8595	13.8040
1.3212	0.1264

```
30.8442           52.2018
10.3347            5.6124
12.3023           20.5187
 0.3668            0.9498
30.9493           13.7672
 2.9821            2.6733
 2.5935            0.2543
 0.6682            1.9840
 5.9755            9.4744
 0.0890            4.4263
 4.7010            2.3728
14.2292            9.6116
 0.8943            1.3058
 1.3961            1.4449
 5.8583            0.8060
 0.0174            2.2955
15.3988           17.5098
100.3884          82.4811
150.7083         147.6452
10.7323           20.6138
 1.2001            4.1400
 4.6595            3.5877
33.5495           54.0743
 0.2894            0.3950
 0.7210            3.9191
48.2809           23.7523
 3.0517            3.2417
 0.5713            0.0166
 7.0136           13.0412
18.5664            7.2417
19.1313           16.0424
14.9465            5.3290
 0.9669            0.0136
867.0612         707.2908
47.9280           37.6687
 0.8946            0.3658
 4.1980            5.6095
47.4987           43.4578
 0.1041            0.0038
431.5750         329.0273
 0.2202            1.8615
 1.9557            8.5125
 0.3472            7.9701
17.1837           16.0415
 9.2588           20.8928
 0.7249            0.0824
12.4961           19.7446
 0.2006            0.5149
 3.3543            6.1065
14.4457           24.5683
20.5040           21.9177
 3.1268            0.4827
 0.0327            0.1633
 2.4052            3.4385
 0.4784            1.4528
```

Conclusions

1.Mean sqaured error of SGD regressor is higher than that of linear regression

2.Most of the SGD weights are higher than that of linear model's weights