In [282]:
```
import pandas as pd
from sklearn.datasets import load_boston
boston = load_boston()
print(boston.data.shape)
print(boston.feature_names)
print(boston.target)
bos = pd.DataFrame(boston.data)
```

```
(506, 13)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATI
O'
 'B' 'LSTAT']
[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]
```

Standardisation

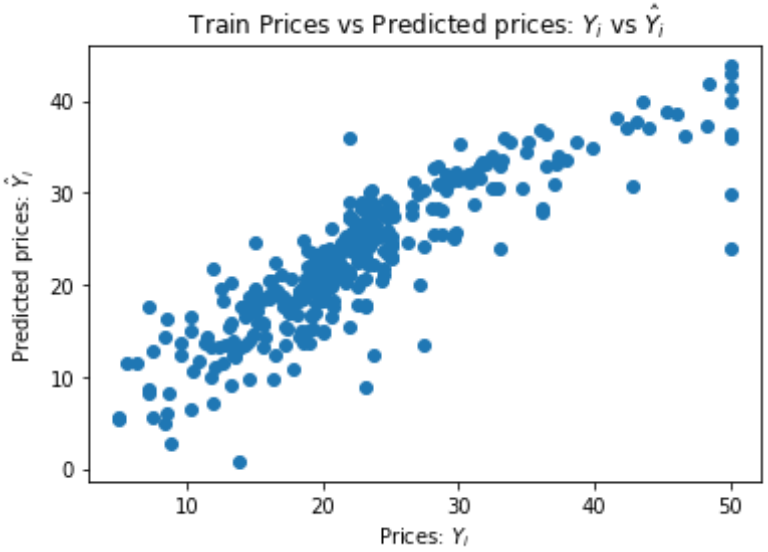
In [283]:
```
bos['PRICE'] = boston.target
X = bos.drop('PRICE', axis = 1)
Y = bos['PRICE']
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler = scaler.fit(X.values)
X = scaler.transform(X.values)
Y= Y.values
```

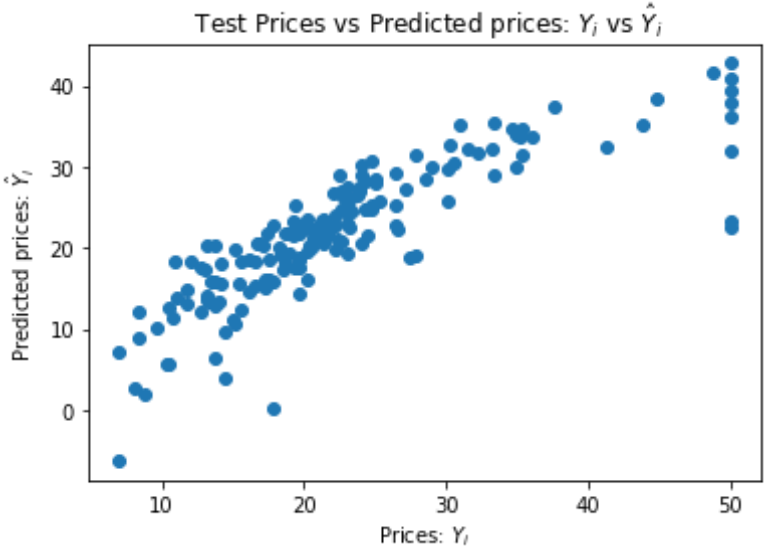
In [284]:
```
from sklearn import cross_validation
X_train, X_test, Y_train, Y_test = cross_validation.train_test_split(X,
 Y, test_size = 0.33, random_state = 5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

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

In [285]:
```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
lm = LinearRegression(fit_intercept=True)
lm.fit(X_train, Y_train)

Y_pred_tr = lm.predict(X_train)
from sklearn.metrics import mean_squared_error
mse1 = mean_squared_error(Y_train,Y_pred_tr)
print("Train Root mean squared error", np.sqrt(mse1))

Y_pred = lm.predict(X_test)
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_test,Y_pred)
print("Test Root mean squared error", np.sqrt(mse))

plt.scatter(Y_train, Y_pred_tr)
plt.xlabel("Prices: $Y_i$")
plt.ylabel("Predicted prices: $\hat{Y}_i$")
plt.title(" Train Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$")
plt.show()

plt.scatter(Y_test, Y_pred)
plt.xlabel("Prices: $Y_i$")
plt.ylabel("Predicted prices: $\hat{Y}_i$")
plt.title(" Test Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$")
plt.show()
```

```
Train Root mean squared error 4.42117161774282
Test Root mean squared error 5.342412121468942
```





Linear Regression Using SKlearn

In [291]:
```
from sklearn.metrics.pairwise import euclidean_distances
dim = 13
def DL_DW(X, error , N):
    b=X.T.dot(error)

    a=(-2/N)*b# here we are finding dl/dw ... which is (-2/N)* x(summat
ion of (yi - wixi -b))

    return a
def DL_DB(error,N):
    return (-2/N)*np.sum(error)#dl/db is (-2/N)* (summation of (yi - wi
xi -b))
def SGD(learning_rate ,n_iterations ,batch_size,decay ):
    N= batch_size
    initial_w = np.random.randn(dim,1)# for first iteration we will tak
e the random value of w as (13,1)dimentions
    initial_b = np.random.randn()#b is same as w0
    ws = [initial_w]#we are first initializing ws with initial w
    bs = [initial_b]#initializing with initial b
    optimal_w =[]#at last our aim is to find the optimal w and optimal
 b
    optimal_b = []
    for i in range(n_iterations):
        indices = np.random.choice(len(X_train),batch_size,replace = Fa
lse)#we are only giving random  indices .. as batch size =10 we will gi
ve 10 random inces
        #Y_pred = []
        #print(indices)
        Y_pred=((X_train[indices].dot(ws[i]))+bs[i])#ypred = X.W +w0


        Y_train1=Y_train[indices].reshape(N,1)

        #rint(Y_train1.shape)
        error=Y_train1-Y_pred
       #print(error.shape)


        #dl_dw =#for sgd    applying w(i+1)=w(i) - r(dl/dw)
```

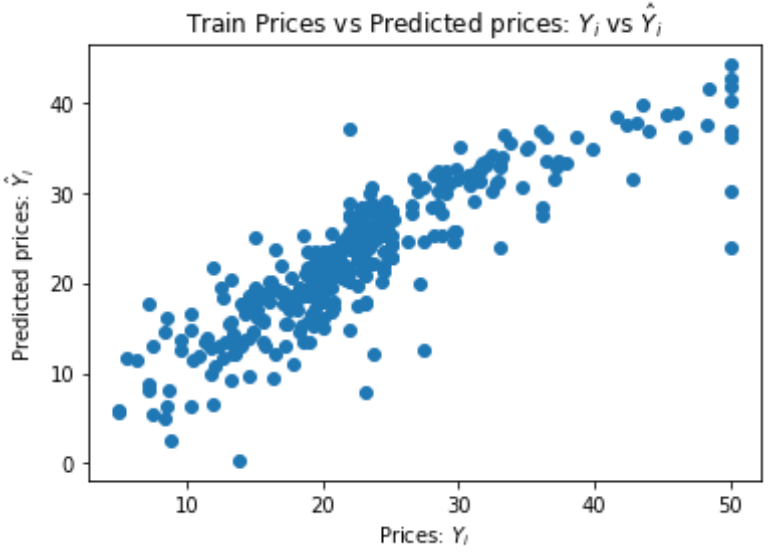
```
        #dl_db =#b(i+1)=b(i) - r (dl/db)

        w = ws[i]-((learning_rate)* DL_DW(X_train[indices],error,N))
        b = bs[i]-((learning_rate)* DL_DB(error,N))
        #print(w , " ", b)
        ws.append(np.copy(w))
        bs.append(np.copy(b))
       #if(ws[i+1]==ws[i]).all() and bs[i+1] == bs[i]:#checking if w is
 optimal or not
        optimal_w = ws[i+1]
        optimal_b = bs[i+1]
            #reak             #if it is optimal break and return optima
l w and b
       #learning_rate = learning_rate * decay
    return optimal_w,optimal_b
```

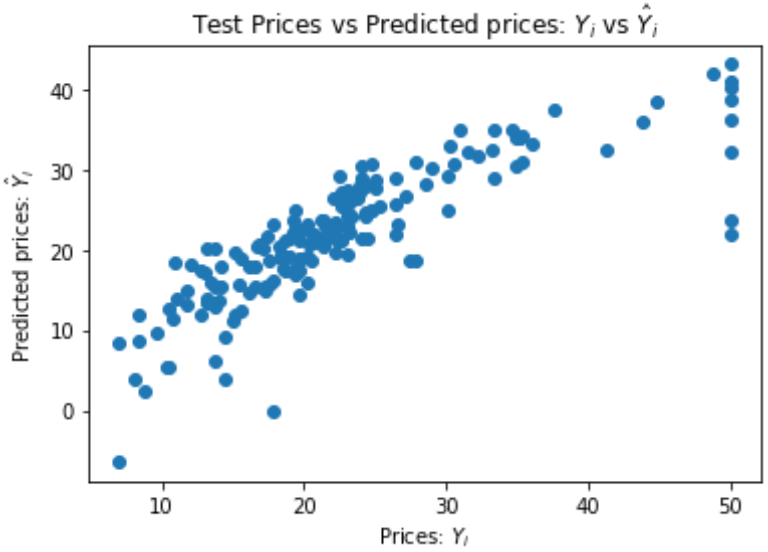
In [292]:
```
w,b=SGD(learning_rate = 0.01,n_iterations =1000,batch_size=100,decay =
0.5)
```

In [293]:
```
from sklearn.metrics import mean_squared_error
Y_train_sgd = (X_train.dot(w)+b)
Y_pred_sgd=(X_test.dot(w))+b
mse1 = mean_squared_error(Y_train,Y_train_sgd)
print("Train Root mean squared error", np.sqrt(mse1))

mse = mean_squared_error(Y_test,Y_pred_sgd)

print("Test Root mean squared error", np.sqrt(mse))


plt.scatter(Y_train, Y_train_sgd)
plt.xlabel("Prices: $Y_i$")
plt.ylabel("Predicted prices: $\hat{Y}_i$")
plt.title("Train Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$")
plt.show()

plt.scatter(Y_test, Y_pred_sgd)
plt.xlabel("Prices: $Y_i$")
```

```
plt.ylabel("Predicted prices: $\hat{Y}_i$")
plt.title("Test Prices vs Predicted prices: $Y_i$ vs $\hat{Y}_i$")
plt.show()
```

```
Train Root mean squared error 4.435356549558849
Test Root mean squared error 5.319809966033083
```





In [295]:
```
from prettytable import PrettyTable
x = PrettyTable(["Table", "Sklearn linear regression", "SGD  on Linear
 regression"])
while True:
    #- Get value
    prompt = input("Please add a head to the list\n")

    try:
         #- Type Casting.
        prompt1 = float(input("Please add a Sklearn linear regression t
o the list\n"))
        prompt2 = float(input("Please enter a SGD  on Linear regression
 for the service\n"))

    except ValueError:
        print("Please enter valid type")
        continue
    #- Add row
    x.add_row([ prompt,prompt1, prompt2])
    #- Ask user to Continue or not.
    choice = input("Continue yes/ no:").lower()
```

```
    if not(choice=="yes" or choice=="y"):
        break
```

```
Please add a head to the list
Train error
Please add a Sklearn linear regression to the list
4.4211
Please enter a SGD  on Linear regression for the service
4.43
Continue yes/ no:y
Please add a head to the list
Test error
Please add a Sklearn linear regression to the list
5.34
Please enter a SGD  on Linear regression for the service
5.31
Continue yes/ no:n
```

In [296]:
```
print(x)
```

```
+-------------+---------------------------+---------------------------+
|    Table    | Sklearn linear regression | SGD  on Linear regression |
+-------------+---------------------------+---------------------------+
| Train error |           4.4211          |            4.43           |
|  Test error |            5.34           |            5.31           |
+-------------+---------------------------+---------------------------+
```