In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
from sklearn.datasets import load_boston
```

In [3]:

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

In [4]:

```
X
```

Out[4]:

```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
         4.9800e+00],
        [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
         9.1400e+00],
        [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
         4.0300e+00],
        ...,
        [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         5.6400e+00],
        [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
         6.4800e+00],
        [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         7.8800e+00]])
```

In [5]:

```
X.shape
```

Out[5]:

```
(506, 13)
```

In [6]:

```
Y
```

Out[6]:

```
array([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])
```

In [7]:

```
Y.shape
```

Out[7]:

```
(506,)
```

In [8]:

```
from sklearn.preprocessing import StandardScaler
```

In [9]:

```
scalar = StandardScaler()
X = scalar.fit_transform(X)
```

In [10]:

```
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error
```

Using Sklearn

In [11]:

```
clf = SGDRegressor()
clf.fit(X,Y)
```

```
c:\users\sahil\appdata\local\programs\python\python36\lib\site-packages\sklearn\linear_model\stochastic
_gradient.py:166: FutureWarning: max_iter and tol parameters have been added in SGDRegressor in 0.19. I
f both are left unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults t
o max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

Out[11]:

```
SGDRegressor(alpha=0.0001, average=False, early_stopping=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, n_iter_no_change=5, penalty='l2', power_t=0.25,
       random_state=None, shuffle=True, tol=None, validation_fraction=0.1,
       verbose=0, warm_start=False)
```

In [12]:

```
mean_squared_error(Y,clf.predict(X))
```

Out[12]:

```
22.8706208219942
```

In [13]:

```
_W1 = clf.coef_
_B1 = clf.intercept_
```

In [14]:

```
_W1
```

Out[14]:

```
array([-0.63853518,  0.57824102, -0.3228565 ,  0.81611229, -1.08736787,
        3.10088725, -0.01703836, -2.10111646,  0.99486493, -0.36523849,
       -1.8556728 ,  0.83868494, -3.40711091])
```

In [15]:

```
_B1
```

Out[15]:

```
array([22.38332442])
```

Custom GD

In [16]:

```
n_iter = 10000
r = 0.01
N = X.shape[0]
W = np.random.normal(size=13)
B = np.random.normal(size=1)
```

In [17]:

```
W
```

Out[17]:

```
array([ 0.34249527,  1.56238351, -0.21374916, -0.24688539,  0.37835649,
       -1.57014716, -0.48728497, -0.68454262,  0.98554942,  0.48503418,
        0.01211782,  0.72440424,  0.09243923])
```

In [18]:

```
B
```

Out[18]:

```
array([0.63127024])
```

In [19]:

```
N
```

Out[19]:

```
506
```

In [20]:

```
predict_value = X.dot(W) - B
predict_value.shape
```

Out[20]:

```
(506,)
```

In [21]:

```
error = Y - predict_value
error.shape
```

Out[21]:

```
(506,)
```

In [22]:

```
_sum_w = (-2) * X.T.dot(error)
_sum_w.shape
```

Out[22]:

```
(13,)
```

In [23]:

```
_sum_b = (-2) * np.sum(error)
_sum_b
```

Out[23]:

```
-23442.04548184834
```

In [24]:

```
while n_iter > 0:
    n_iter -= 1
    predict_value = X.dot(W) + B
    error = Y - predict_value
    if n_iter % 1000 == 0:
        print('Epoch:{0}, MSE:{1}'.format(10000-n_iter,mean_squared_error(Y,predict_value)))
    _sum_w = (-2) * X.T.dot(error)
    _sum_b = (-2) * np.sum(error)
    W = W - r * _sum_w / N
    B = B - r * _sum_b / N
```

```
Epoch:1000, MSE:21.941804634309364
Epoch:2000, MSE:21.89841451941127
Epoch:3000, MSE:21.895113097524177
Epoch:4000, MSE:21.894853371173934
Epoch:5000, MSE:21.89483292826008
Epoch:6000, MSE:21.894831319198637
```

```
Epoch:7000, MSE:21.894831192549418
Epoch:8000, MSE:21.894831182580862
Epoch:9000, MSE:21.89483118179624
Epoch:10000, MSE:21.89483118173448
```

In [25]:

```
mean_squared_error(Y, predict_value)
```

Out[25]:

```
21.89483118173448
```

In [26]:

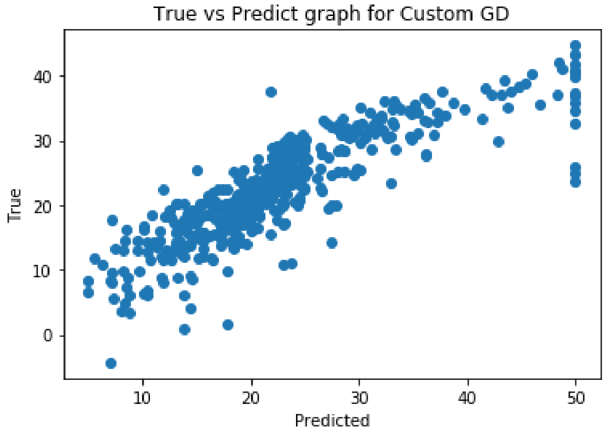
```
import matplotlib.pyplot as plt
```

In [27]:

```
predict_value = X.dot(W) + B
```

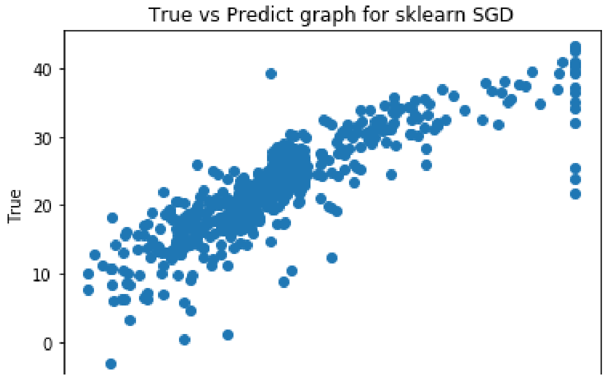
In [28]:

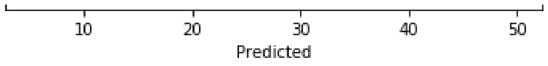
```
plt.scatter(Y, predict_value)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('True vs Predict graph for Custom GD')
plt.show()
```



In [29]:

```
plt.scatter(Y,clf.predict(X))
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('True vs Predict graph for sklearn SGD')
plt.show()
```





In [30]:

```
from prettytable import PrettyTable
```

In [31]:

```
x = PrettyTable()

x.field_names = ["Model", "Weights", "Intercept", "MSE"]

x.add_row(["sklearn", _W1, _B1, mean_squared_error(Y,clf.predict(X))])
x.add_row(["custom", W, B, mean_squared_error(Y,predict_value)])

print(x)
```

```
+---------+--------------------------------------------------------------------------+---------------+-
-------------------+
|  Model  |                                 Weights                                  |   Intercept   |
MSE        |
+---------+--------------------------------------------------------------------------+---------------+-
-------------------+
| sklearn | [-0.63853518  0.57824102 -0.3228565   0.81611229 -1.08736787  3.10088725 | [22.38332442] |
22.8706208219942 |
|         |  -0.01703836 -2.10111646  0.99486493 -0.36523849 -1.8556728   0.83868494 |               |
|
|         |                               -3.40711091]                               |               |
|
|  custom | [-0.92814565  1.08156789  0.14089771  0.68174005 -2.05671787  2.67423058 | [22.53280632] |
21.89483118173447 |
|         |   0.01946572 -3.10404442  2.66221188 -2.07677513 -2.06060645  0.84926838 |               |
|
|         |                               -3.7436269 ]                               |               |
|
+---------+--------------------------------------------------------------------------+---------------+-
-------------------+
```

In [ ]: