Linear Regression on Boston dataset

In [137]:

```
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error
```

In [101]:

```
from sklearn.datasets import load_boston
boston = load_boston()
```

In [102]:

```
print(boston.data.shape)
```

```
(506, 13)
```

In [103]:

```
print(boston.feature_names)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

In [22]:

```
print(boston.target)
```

```
[ 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 [4]:

```python
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually th
e target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collin
earity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth
International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kau
fmann.
```

In [108]:

```python
#Loading the Boston data and splitting into test and train.
boston_data=pd.DataFrame(load_boston().data,columns=load_boston().feature_names)
Y=load_boston().target
```

```
X=load_boston().data
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)
```

In [110]:

```
# standardizing data
scaler = StandardScaler().fit(x_train)
x_train = scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

In [111]:

```
train_data=pd.DataFrame(x_train)
train_data['PRICE']=y_train
train_data.head(3)

x_test=np.array(x_test)
y_test=np.array(y_test)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(354, 13)
(152, 13)
(354,)
(152,)
```

Linear Regression using inbuilt function

In [141]:

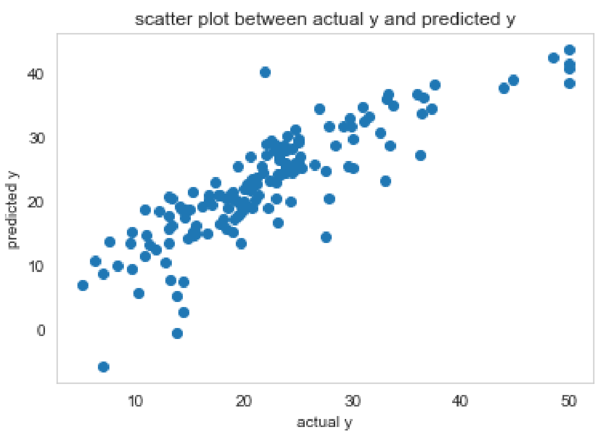
```
# code source:https://medium.com/@haydar_ai/learning-data-science-day-9-linear-regression-on-boston-hou
sing-dataset-cd62a80775ef
from sklearn.linear_model import LinearRegression

lm = LinearRegression()
lm.fit(x_train, y_train)

Y_pred = lm.predict(x_test)

plot_(y_test,Y_pred)

#plt.scatter(test_data, y_pred)
#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()
```




```
Mean Squared Error between Actual and Predicted values:  22.615007025836285
```

Out[141]:

22.615007025836285

Using SGD Regressor

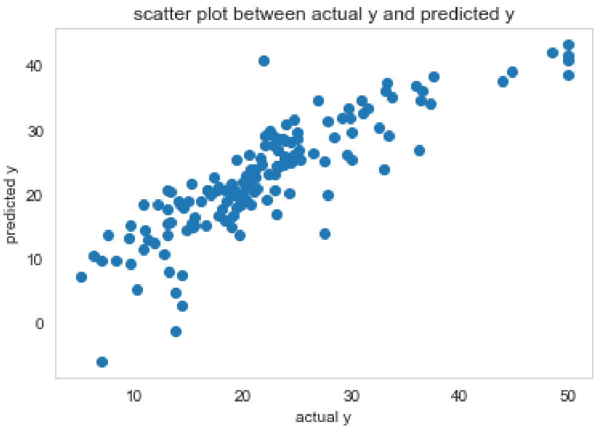
In [165]:

```
# code source:https://medium.com/@haydar_ai/learning-data-science-day-9-linear-regression-on-boston-hou
sing-dataset-cd62a80775ef
from sklearn.linear_model import SGDRegressor

lm = SGDRegressor()
lm.fit(x_train, y_train)

Y_pred = lm.predict(x_test)

plot_(y_test,Y_pred)
```



Mean Squared Error between Actual and Predicted values: 22.88629594848648

Out[165]:

22.88629594848648

In [177]:

```
X_train.shape[0]
X_train.shape[1]
```

Out[177]:

13

Computing SDG

In [206]:

```
def Compute_SGD(X_train, y_train, lr_rate, n_epochs):
    w_coeff=np.random.randn(13,1) #Randomly initalizing weights
    b_coeff=np.random.randn(1,1)  #Randomly picking up intercept value.

    for epoch in range(1,n_epochs+1):
        sum_errors = 0 #Sum of squared loss.
        N = X_train.shape[0] #The variable N in the SGD equation.(339 in this case)

        for i in range(N):
            batch_size = np.random.randint(0,N)  # randomly picks one value from 0 to 339
            X_i = X_train[batch_size,:].reshape(1,X_train.shape[1])# Reshaping to size(1,13)
            y_i = y_train[batch_size].reshape(1,1)

            y_pred = np.dot(X_i,w_coeff) + b_coeff                 #y_true = WT.X + B(not taking trans
pose because we initialised w as (13,1))
            loss = y_pred - y_i                                    #w_grad = df/dw = (-2/N)*(X)*(y-(WT
.X+B)) = (-2/N)*(X) * loss
```

```
.X+B)) = (-2/N)*(X) * loss
            sum_errors += loss**2                                   #b_grad = df/db = (-2/N)*(y-(WT.X+B
)) = (-2/N) * loss

            w_grad = X_i.T.dot((y_pred - y_i))
            b_grad = (y_pred - y_i)

            w_coeff = w_coeff -(2/N)*lr_rate*(w_grad)              #wold = wnew - learning rate * df/d
w
            b_coeff = b_coeff - (2/N)*lr_rate*(b_grad)             #bold = bnew - learning rate * df/d
b

        #print("Epoch: %d, Loss: %.3f" %(epoch, sum_errors/N))



    return w_coeff, b_coeff


def predict(X_test, w_coeff, b_coeff):
    X_test=np.array(X_test)
    y_pred =[]
    for i in range(0,len(X_test)):
        y=np.asscalar(np.dot(w_coeff,X_test[i]) + b_coeff) #Convert an array of size 1 to its scalar eq
uivalent.
        y_pred.append(y)
    return np.array(y_pred)


def plot_scatter(y_test,y_pred):
    plt.scatter(y_test,y_pred)
    plt.title('Scatter plot between Actual and Predicted Y.')
    plt.xlabel('Actual Y')
    plt.ylabel('Predicted Y')
    plt.grid(b=True, linewidth=0.5)
    plt.show()

    #Get the mean squared error between the predicted and the actual values.
    mse=mean_squared_error(y_test,y_pred)
    print('Mean Squared Error between Actual and Predicted values: ',mse)
    return mse
```

In [207]:

```
#Get the optimal value of the w coefficients and b coefficients.
w_coeff_optimal, b_coeff_optimal = Compute_SGD(x_train, y_train, lr_rate=0.01,n_epochs=1000)
```

In [169]:

```
#Get the optimal value of the w coefficients and b coefficients.
w_coeff_optimal, b_coeff_optimal = Compute_SGD(x_train, y_train, lr_rate=0.01,n_epochs=500)
```

```
Epoch: 1, Loss: 580.250
Epoch: 2, Loss: 582.005
Epoch: 3, Loss: 513.077
Epoch: 4, Loss: 516.218
Epoch: 5, Loss: 517.459
Epoch: 6, Loss: 447.389
Epoch: 7, Loss: 491.677
Epoch: 8, Loss: 460.197
Epoch: 9, Loss: 402.876
Epoch: 10, Loss: 406.495
Epoch: 11, Loss: 388.413
Epoch: 12, Loss: 388.862
Epoch: 13, Loss: 355.457
Epoch: 14, Loss: 345.264
Epoch: 15, Loss: 312.179
Epoch: 16, Loss: 321.965
Epoch: 17, Loss: 301.958
Epoch: 18, Loss: 289.002
Epoch: 19, Loss: 278.762
Epoch: 20, Loss: 259.681
Epoch: 21, Loss: 248.049
Epoch: 22, Loss: 232.697
```

```
Epoch: 23, Loss: 260.703
Epoch: 24, Loss: 227.762
Epoch: 25, Loss: 190.353
Epoch: 26, Loss: 214.358
Epoch: 27, Loss: 244.215
Epoch: 28, Loss: 195.348
Epoch: 29, Loss: 208.377
Epoch: 30, Loss: 175.595
Epoch: 31, Loss: 191.412
Epoch: 32, Loss: 172.517
Epoch: 33, Loss: 151.837
Epoch: 34, Loss: 175.240
Epoch: 35, Loss: 196.071
Epoch: 36, Loss: 169.896
Epoch: 37, Loss: 145.891
Epoch: 38, Loss: 144.091
Epoch: 39, Loss: 129.340
Epoch: 40, Loss: 122.267
Epoch: 41, Loss: 137.537
Epoch: 42, Loss: 119.302
Epoch: 43, Loss: 120.910
Epoch: 44, Loss: 110.798
Epoch: 45, Loss: 109.677
Epoch: 46, Loss: 104.522
Epoch: 47, Loss: 97.461
Epoch: 48, Loss: 114.881
Epoch: 49, Loss: 98.292
Epoch: 50, Loss: 90.804
Epoch: 51, Loss: 109.521
Epoch: 52, Loss: 80.235
Epoch: 53, Loss: 81.536
Epoch: 54, Loss: 94.678
Epoch: 55, Loss: 92.248
Epoch: 56, Loss: 94.076
Epoch: 57, Loss: 91.469
Epoch: 58, Loss: 68.678
Epoch: 59, Loss: 83.356
Epoch: 60, Loss: 79.048
Epoch: 61, Loss: 82.121
Epoch: 62, Loss: 75.078
Epoch: 63, Loss: 57.695
Epoch: 64, Loss: 69.557
Epoch: 65, Loss: 75.094
Epoch: 66, Loss: 61.915
Epoch: 67, Loss: 65.830
Epoch: 68, Loss: 62.460
Epoch: 69, Loss: 76.504
Epoch: 70, Loss: 57.428
Epoch: 71, Loss: 64.230
Epoch: 72, Loss: 63.500
Epoch: 73, Loss: 65.960
Epoch: 74, Loss: 62.277
Epoch: 75, Loss: 57.062
Epoch: 76, Loss: 40.188
Epoch: 77, Loss: 59.136
Epoch: 78, Loss: 44.006
Epoch: 79, Loss: 51.672
Epoch: 80, Loss: 49.465
Epoch: 81, Loss: 40.094
Epoch: 82, Loss: 42.968
Epoch: 83, Loss: 43.489
Epoch: 84, Loss: 47.108
Epoch: 85, Loss: 33.057
Epoch: 86, Loss: 72.227
Epoch: 87, Loss: 32.950
Epoch: 88, Loss: 32.315
Epoch: 89, Loss: 52.168
Epoch: 90, Loss: 35.871
Epoch: 91, Loss: 34.373
Epoch: 92, Loss: 44.056
Epoch: 93, Loss: 25.462
Epoch: 94, Loss: 34.085
Epoch: 95, Loss: 38.847
Epoch: 96, Loss: 44.478
Epoch: 97, Loss: 43.347
Epoch: 98, Loss: 53.638
Epoch: 99, Loss: 34.697
```

```
Epoch: 100, Loss: 31.395
Epoch: 101, Loss: 37.627
Epoch: 102, Loss: 30.024
Epoch: 103, Loss: 38.358
Epoch: 104, Loss: 42.801
Epoch: 105, Loss: 37.405
Epoch: 106, Loss: 41.459
Epoch: 107, Loss: 37.687
Epoch: 108, Loss: 31.302
Epoch: 109, Loss: 34.069
Epoch: 110, Loss: 37.039
Epoch: 111, Loss: 20.108
Epoch: 112, Loss: 41.966
Epoch: 113, Loss: 32.914
Epoch: 114, Loss: 28.809
Epoch: 115, Loss: 29.728
Epoch: 116, Loss: 21.795
Epoch: 117, Loss: 30.997
Epoch: 118, Loss: 29.589
Epoch: 119, Loss: 26.815
Epoch: 120, Loss: 28.886
Epoch: 121, Loss: 31.854
Epoch: 122, Loss: 25.372
Epoch: 123, Loss: 31.352
Epoch: 124, Loss: 29.691
Epoch: 125, Loss: 23.183
Epoch: 126, Loss: 24.298
Epoch: 127, Loss: 34.777
Epoch: 128, Loss: 21.619
Epoch: 129, Loss: 29.404
Epoch: 130, Loss: 25.064
Epoch: 131, Loss: 29.584
Epoch: 132, Loss: 21.983
Epoch: 133, Loss: 27.986
Epoch: 134, Loss: 23.882
Epoch: 135, Loss: 29.104
Epoch: 136, Loss: 24.300
Epoch: 137, Loss: 24.257
Epoch: 138, Loss: 19.450
Epoch: 139, Loss: 38.450
Epoch: 140, Loss: 25.566
Epoch: 141, Loss: 31.958
Epoch: 142, Loss: 22.450
Epoch: 143, Loss: 27.755
Epoch: 144, Loss: 34.399
Epoch: 145, Loss: 23.961
Epoch: 146, Loss: 26.594
Epoch: 147, Loss: 29.964
Epoch: 148, Loss: 35.921
Epoch: 149, Loss: 25.359
Epoch: 150, Loss: 36.429
Epoch: 151, Loss: 21.508
Epoch: 152, Loss: 28.057
Epoch: 153, Loss: 28.858
Epoch: 154, Loss: 35.008
Epoch: 155, Loss: 23.511
Epoch: 156, Loss: 22.164
Epoch: 157, Loss: 26.447
Epoch: 158, Loss: 34.067
Epoch: 159, Loss: 23.725
Epoch: 160, Loss: 28.582
Epoch: 161, Loss: 26.376
Epoch: 162, Loss: 30.014
Epoch: 163, Loss: 22.696
Epoch: 164, Loss: 28.324
Epoch: 165, Loss: 22.465
Epoch: 166, Loss: 28.231
Epoch: 167, Loss: 34.697
Epoch: 168, Loss: 28.261
Epoch: 169, Loss: 21.333
Epoch: 170, Loss: 31.330
Epoch: 171, Loss: 22.840
Epoch: 172, Loss: 20.041
Epoch: 173, Loss: 24.122
Epoch: 174, Loss: 28.430
Epoch: 175, Loss: 20.468
Epoch: 176, Loss: 25.512
```

```
Epoch: 177, Loss: 16.497
Epoch: 178, Loss: 23.872
Epoch: 179, Loss: 28.131
Epoch: 180, Loss: 24.374
Epoch: 181, Loss: 24.384
Epoch: 182, Loss: 20.601
Epoch: 183, Loss: 22.733
Epoch: 184, Loss: 20.879
Epoch: 185, Loss: 24.715
Epoch: 186, Loss: 21.824
Epoch: 187, Loss: 15.664
Epoch: 188, Loss: 25.991
Epoch: 189, Loss: 29.544
Epoch: 190, Loss: 25.228
Epoch: 191, Loss: 27.701
Epoch: 192, Loss: 32.023
Epoch: 193, Loss: 21.271
Epoch: 194, Loss: 23.418
Epoch: 195, Loss: 32.667
Epoch: 196, Loss: 25.890
Epoch: 197, Loss: 18.609
Epoch: 198, Loss: 18.234
Epoch: 199, Loss: 20.476
Epoch: 200, Loss: 26.115
Epoch: 201, Loss: 23.364
Epoch: 202, Loss: 28.418
Epoch: 203, Loss: 17.036
Epoch: 204, Loss: 30.880
Epoch: 205, Loss: 24.832
Epoch: 206, Loss: 20.310
Epoch: 207, Loss: 22.867
Epoch: 208, Loss: 25.213
Epoch: 209, Loss: 26.449
Epoch: 210, Loss: 20.809
Epoch: 211, Loss: 21.607
Epoch: 212, Loss: 22.841
Epoch: 213, Loss: 25.345
Epoch: 214, Loss: 21.596
Epoch: 215, Loss: 28.063
Epoch: 216, Loss: 20.227
Epoch: 217, Loss: 19.886
Epoch: 218, Loss: 23.913
Epoch: 219, Loss: 24.465
Epoch: 220, Loss: 32.422
Epoch: 221, Loss: 25.559
Epoch: 222, Loss: 20.546
Epoch: 223, Loss: 24.395
Epoch: 224, Loss: 30.980
Epoch: 225, Loss: 21.415
Epoch: 226, Loss: 25.790
Epoch: 227, Loss: 17.456
Epoch: 228, Loss: 27.357
Epoch: 229, Loss: 22.620
Epoch: 230, Loss: 18.918
Epoch: 231, Loss: 23.380
Epoch: 232, Loss: 33.037
Epoch: 233, Loss: 15.769
Epoch: 234, Loss: 20.933
Epoch: 235, Loss: 18.481
Epoch: 236, Loss: 27.654
Epoch: 237, Loss: 28.795
Epoch: 238, Loss: 17.232
Epoch: 239, Loss: 23.050
Epoch: 240, Loss: 19.929
Epoch: 241, Loss: 25.574
Epoch: 242, Loss: 22.655
Epoch: 243, Loss: 24.544
Epoch: 244, Loss: 20.277
Epoch: 245, Loss: 20.303
Epoch: 246, Loss: 26.077
Epoch: 247, Loss: 18.283
Epoch: 248, Loss: 18.355
Epoch: 249, Loss: 33.214
Epoch: 250, Loss: 15.000
Epoch: 251, Loss: 31.695
Epoch: 252, Loss: 21.522
Epoch: 253, Loss: 22.963
```

```
Epoch: 254, Loss: 25.910
Epoch: 255, Loss: 17.254
Epoch: 256, Loss: 21.508
Epoch: 257, Loss: 20.574
Epoch: 258, Loss: 26.253
Epoch: 259, Loss: 23.695
Epoch: 260, Loss: 26.077
Epoch: 261, Loss: 28.076
Epoch: 262, Loss: 22.325
Epoch: 263, Loss: 15.485
Epoch: 264, Loss: 16.916
Epoch: 265, Loss: 23.672
Epoch: 266, Loss: 32.321
Epoch: 267, Loss: 25.005
Epoch: 268, Loss: 29.624
Epoch: 269, Loss: 19.829
Epoch: 270, Loss: 21.688
Epoch: 271, Loss: 25.772
Epoch: 272, Loss: 25.612
Epoch: 273, Loss: 21.662
Epoch: 274, Loss: 20.662
Epoch: 275, Loss: 31.166
Epoch: 276, Loss: 20.158
Epoch: 277, Loss: 18.035
Epoch: 278, Loss: 24.739
Epoch: 279, Loss: 19.607
Epoch: 280, Loss: 22.665
Epoch: 281, Loss: 20.993
Epoch: 282, Loss: 22.984
Epoch: 283, Loss: 24.583
Epoch: 284, Loss: 26.122
Epoch: 285, Loss: 20.900
Epoch: 286, Loss: 22.046
Epoch: 287, Loss: 22.624
Epoch: 288, Loss: 23.677
Epoch: 289, Loss: 24.950
Epoch: 290, Loss: 25.429
Epoch: 291, Loss: 26.436
Epoch: 292, Loss: 21.655
Epoch: 293, Loss: 20.779
Epoch: 294, Loss: 29.573
Epoch: 295, Loss: 24.971
Epoch: 296, Loss: 22.987
Epoch: 297, Loss: 37.409
Epoch: 298, Loss: 23.919
Epoch: 299, Loss: 25.158
Epoch: 300, Loss: 21.253
Epoch: 301, Loss: 25.823
Epoch: 302, Loss: 22.232
Epoch: 303, Loss: 16.287
Epoch: 304, Loss: 24.911
Epoch: 305, Loss: 25.666
Epoch: 306, Loss: 27.875
Epoch: 307, Loss: 30.976
Epoch: 308, Loss: 23.612
Epoch: 309, Loss: 20.843
Epoch: 310, Loss: 18.323
Epoch: 311, Loss: 28.681
Epoch: 312, Loss: 20.882
Epoch: 313, Loss: 26.109
Epoch: 314, Loss: 28.073
Epoch: 315, Loss: 22.580
Epoch: 316, Loss: 18.500
Epoch: 317, Loss: 24.122
Epoch: 318, Loss: 23.023
Epoch: 319, Loss: 26.410
Epoch: 320, Loss: 28.036
Epoch: 321, Loss: 24.861
Epoch: 322, Loss: 25.736
Epoch: 323, Loss: 25.139
Epoch: 324, Loss: 24.611
Epoch: 325, Loss: 25.177
Epoch: 326, Loss: 20.816
Epoch: 327, Loss: 31.609
Epoch: 328, Loss: 24.829
Epoch: 329, Loss: 27.323
Epoch: 330, Loss: 20.557
```

```
Epoch: 331, Loss: 24.552
Epoch: 332, Loss: 18.046
Epoch: 333, Loss: 33.088
Epoch: 334, Loss: 28.549
Epoch: 335, Loss: 26.586
Epoch: 336, Loss: 19.876
Epoch: 337, Loss: 25.399
Epoch: 338, Loss: 20.654
Epoch: 339, Loss: 24.675
Epoch: 340, Loss: 13.317
Epoch: 341, Loss: 19.158
Epoch: 342, Loss: 27.887
Epoch: 343, Loss: 23.729
Epoch: 344, Loss: 25.471
Epoch: 345, Loss: 33.896
Epoch: 346, Loss: 21.267
Epoch: 347, Loss: 19.149
Epoch: 348, Loss: 20.797
Epoch: 349, Loss: 25.687
Epoch: 350, Loss: 19.729
Epoch: 351, Loss: 25.638
Epoch: 352, Loss: 23.308
Epoch: 353, Loss: 25.700
Epoch: 354, Loss: 25.330
Epoch: 355, Loss: 20.121
Epoch: 356, Loss: 22.873
Epoch: 357, Loss: 23.574
Epoch: 358, Loss: 26.577
Epoch: 359, Loss: 22.688
Epoch: 360, Loss: 19.042
Epoch: 361, Loss: 24.041
Epoch: 362, Loss: 26.685
Epoch: 363, Loss: 19.412
Epoch: 364, Loss: 16.842
Epoch: 365, Loss: 33.811
Epoch: 366, Loss: 25.613
Epoch: 367, Loss: 24.152
Epoch: 368, Loss: 22.041
Epoch: 369, Loss: 27.458
Epoch: 370, Loss: 12.664
Epoch: 371, Loss: 22.397
Epoch: 372, Loss: 22.996
Epoch: 373, Loss: 26.040
Epoch: 374, Loss: 21.387
Epoch: 375, Loss: 23.011
Epoch: 376, Loss: 22.689
Epoch: 377, Loss: 20.224
Epoch: 378, Loss: 20.532
Epoch: 379, Loss: 17.732
Epoch: 380, Loss: 26.145
Epoch: 381, Loss: 25.552
Epoch: 382, Loss: 23.338
Epoch: 383, Loss: 22.503
Epoch: 384, Loss: 21.757
Epoch: 385, Loss: 24.162
Epoch: 386, Loss: 28.280
Epoch: 387, Loss: 16.025
Epoch: 388, Loss: 18.277
Epoch: 389, Loss: 23.335
Epoch: 390, Loss: 16.677
Epoch: 391, Loss: 24.550
Epoch: 392, Loss: 28.049
Epoch: 393, Loss: 24.567
Epoch: 394, Loss: 18.609
Epoch: 395, Loss: 21.891
Epoch: 396, Loss: 22.569
Epoch: 397, Loss: 18.878
Epoch: 398, Loss: 19.763
Epoch: 399, Loss: 19.608
Epoch: 400, Loss: 23.941
Epoch: 401, Loss: 22.049
Epoch: 402, Loss: 20.445
Epoch: 403, Loss: 24.906
Epoch: 404, Loss: 21.748
Epoch: 405, Loss: 27.777
Epoch: 406, Loss: 27.053
Epoch: 407, Loss: 27.759
```

```
Epoch: 408, Loss: 29.527
Epoch: 409, Loss: 26.365
Epoch: 410, Loss: 22.007
Epoch: 411, Loss: 22.442
Epoch: 412, Loss: 21.077
Epoch: 413, Loss: 17.296
Epoch: 414, Loss: 21.812
Epoch: 415, Loss: 24.708
Epoch: 416, Loss: 22.546
Epoch: 417, Loss: 19.481
Epoch: 418, Loss: 21.829
Epoch: 419, Loss: 21.847
Epoch: 420, Loss: 19.088
Epoch: 421, Loss: 17.839
Epoch: 422, Loss: 20.197
Epoch: 423, Loss: 20.738
Epoch: 424, Loss: 26.576
Epoch: 425, Loss: 19.885
Epoch: 426, Loss: 18.953
Epoch: 427, Loss: 20.378
Epoch: 428, Loss: 17.365
Epoch: 429, Loss: 20.183
Epoch: 430, Loss: 18.058
Epoch: 431, Loss: 21.201
Epoch: 432, Loss: 22.033
Epoch: 433, Loss: 20.634
Epoch: 434, Loss: 21.832
Epoch: 435, Loss: 19.218
Epoch: 436, Loss: 18.444
Epoch: 437, Loss: 27.281
Epoch: 438, Loss: 20.662
Epoch: 439, Loss: 33.056
Epoch: 440, Loss: 25.779
Epoch: 441, Loss: 27.089
Epoch: 442, Loss: 22.963
Epoch: 443, Loss: 15.815
Epoch: 444, Loss: 20.246
Epoch: 445, Loss: 19.708
Epoch: 446, Loss: 26.152
Epoch: 447, Loss: 25.786
Epoch: 448, Loss: 26.139
Epoch: 449, Loss: 22.552
Epoch: 450, Loss: 25.146
Epoch: 451, Loss: 18.351
Epoch: 452, Loss: 19.582
Epoch: 453, Loss: 21.055
Epoch: 454, Loss: 23.230
Epoch: 455, Loss: 18.043
Epoch: 456, Loss: 18.726
Epoch: 457, Loss: 23.595
Epoch: 458, Loss: 23.617
Epoch: 459, Loss: 19.074
Epoch: 460, Loss: 22.473
Epoch: 461, Loss: 20.676
Epoch: 462, Loss: 29.384
Epoch: 463, Loss: 27.057
Epoch: 464, Loss: 27.841
Epoch: 465, Loss: 21.674
Epoch: 466, Loss: 19.931
Epoch: 467, Loss: 16.666
Epoch: 468, Loss: 31.806
Epoch: 469, Loss: 20.190
Epoch: 470, Loss: 18.167
Epoch: 471, Loss: 27.686
Epoch: 472, Loss: 19.191
Epoch: 473, Loss: 23.061
Epoch: 474, Loss: 19.515
Epoch: 475, Loss: 19.689
Epoch: 476, Loss: 23.562
Epoch: 477, Loss: 19.935
Epoch: 478, Loss: 23.186
Epoch: 479, Loss: 20.982
Epoch: 480, Loss: 26.573
Epoch: 481, Loss: 24.998
Epoch: 482, Loss: 25.832
Epoch: 483, Loss: 19.759
Epoch: 484, Loss: 22.314
```

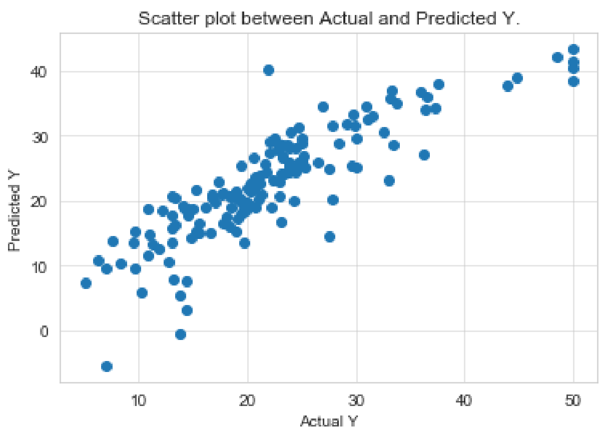
```
Epoch: 485, Loss: 22.712
Epoch: 486, Loss: 23.048
Epoch: 487, Loss: 17.888
Epoch: 488, Loss: 20.866
Epoch: 489, Loss: 21.617
Epoch: 490, Loss: 22.730
Epoch: 491, Loss: 19.196
Epoch: 492, Loss: 13.493
Epoch: 493, Loss: 19.511
Epoch: 494, Loss: 23.404
Epoch: 495, Loss: 20.026
Epoch: 496, Loss: 24.794
Epoch: 497, Loss: 35.766
Epoch: 498, Loss: 25.406
Epoch: 499, Loss: 25.868
Epoch: 500, Loss: 19.415
```

In [208]:

```
y_pred = predict(x_test, w_coeff_optimal.T, b_coeff_optimal)

#Draw the scatter plot
mse1=plot_scatter(y_test,y_pred)
```

```
D:\AAnaconda\lib\site-packages\ipykernel_launcher.py:35: DeprecationWarning: np.asscalar(a) is deprecat
ed since NumPy v1.16, use a.item() instead
```




```
Mean Squared Error between Actual and Predicted values:  22.67705485681849
```

Observation:

In [210]:

```
from prettytable import PrettyTable

x = PrettyTable()

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

x.add_row(["Custom SGD Regression", 22.67])
x.add_row(["Linear Regression", 22.61])
x.add_row(["SGD Regression",22.88])


print(x)
```

```
+-----------------------+-------+
|         Model         |  MSE  |
+-----------------------+-------+
| Custom SGD Regression | 22.67 |
|   Linear Regression   | 22.61 |
|     SGD Regression    | 22.88 |
+-----------------------+-------+
```

Observation: Custom SGD regressor is applied on Boston dataset with learning rate of 0.01 and 1000 epochs. The obtained Mean squared error value 22.67 is almost same as Linear Regression.