https://www.kaggle.com/c/boston-housing

Boston House Prices dataset

Data Set Characteristics:

```
:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical

:Median Value (attribute 14) is usually the 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. http://archive.ics.uci.edu/ml/datasets/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.

In [1]:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cross_validation import train_test_split
import seaborn as sns
import numpy as np
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
%matplotlib inline
```

```
C:\Users\kingsubham27091995\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41:
DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection
module into which all the refactored classes and functions are moved. Also note that the interface
of the new CV iterators are different from that of this module. This module will be removed in 0.2
0.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [2]:

```
# loading boston datasets
from sklearn.datasets import load_boston
import pandas as pd
```

In [3]:

```
# spliting the data into train and test

boston_data=pd.DataFrame(data=load_boston().data)
price_data=load_boston().target
X_train, X_test, y_train, y_test=train_test_split(boston_data, price_data, test_size=0.33, random_s
tate=5)
```

In [4]:

```
# applying column standardization on train and test data
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
scalar=StandardScaler()
X_train=scalar.fit_transform(np.array(X_train))
X_test=scalar.transform(np.array(X_test))
```

In [5]:

```
#preparing training data for manual sgd regressor
manual_train=pd.DataFrame(data=X_train)
manual_train['price']=y_train
```

In [6]:

```
manual_train.head(3)
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	10	11	
0	0.911839	-0.502419	1.072305	-0.256978	1.633548	0.486034	0.962774	-0.823477	1.655334	1.552100	0.808078	-2.842959	1
1	-0.411727	-0.502419	-1.129795	-0.256978	-0.552451	1.028078	0.668619	-0.183274	-0.871371	-0.802704	-0.304174	0.427436	-0
2	0.124583	-0.502419	1.072305	-0.256978	1.441946	-3.913414	0.725324	-1.075955	1.655334	1.552100	0.808078	-0.053353	-0

In [7]:

```
# converting to numpy array, which will be available for both SGDRegressor of sklearn and manual s
gd regressor
X_test=np.array(X_test)
y_test=np.array(y_test)
```

In [8]:

```
results=pd.DataFrame(columns=['S_No', 'Algorithm', 'Alpha', 'LearningRateVariation', 'Learning
Rate', 'Power_t', 'No. of Iterations', 'Mean Squared Error','Weights'])
```

SGDRegressor vs Manual SGD

A) Fixing learning rate(eta0) to 0.01 and Learning Rate Variation='Constant'

SGDRegressor

In [9]:

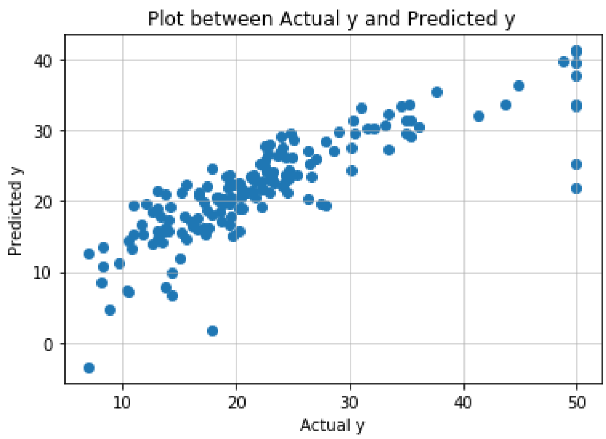
```
#the functioning of this function is to use sklearn SGDRegressor and predict the price
#this function takes alpha, learning rate variation , initial learning rate(eta0), number of itera
tion , power_t, and all test and train data as an argument
#this function returns weight, intercept and mean squared error
def sklearn_sgd(alpha, lr_rate_variation, eta0=0.01, power_t=0.25, n_iter=100, X_train=X_train, X_t
est=X_test, y_train=y_train, y_test=y_test):
    clf=SGDRegressor(alpha=alpha, penalty=None, learning_rate=lr_rate_variation, eta0=eta0, power_t
=power_t, n_iter=n_iter)
    clf.fit(X_train, y_train)
    y_pred=clf.predict(X_test)

    #scatter plot
    plt.scatter(y_test,y_pred)
    plt.title('Plot between Actual y and Predicted y')
    plt.xlabel('Actual y')
    plt.ylabel('Predicted y')
    plt.grid(b=True, linewidth=0.5)
    plt.show()


    ## Mean Squared Error(MSE)
    mse=mean_squared_error(y_test,y_pred)
    print('mean sq error=', mse)
    print('number of iteration=', n_iter)
    print('Weight =',clf.coef_)

    return clf.coef_, clf.intercept_, mse
```

1. n_iter=1, lr_rate=0.01, lr_rate_variation='constant'

In [10]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='constant', eta0=0.01, n_iter=1
)
```

```
C:\Users\kingsubham27091995\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)
```




```
mean sq error= 28.85059286297685
number of iteration= 1
Weight = [-1.15653814  0.66974888 -0.08178764  0.6382138  -0.66797691  2.95215349
 -0.10357986 -1.93735889  1.12825615 -0.2267697  -1.78106711  0.95286871
 -2.90881683]
```

In [11]:

```
new=[1, 'SGDRegressor', 0.0001, 'constant', 0.01, 0.25, 1, error_sgd,w_sgd]
results.loc[0]=new
```

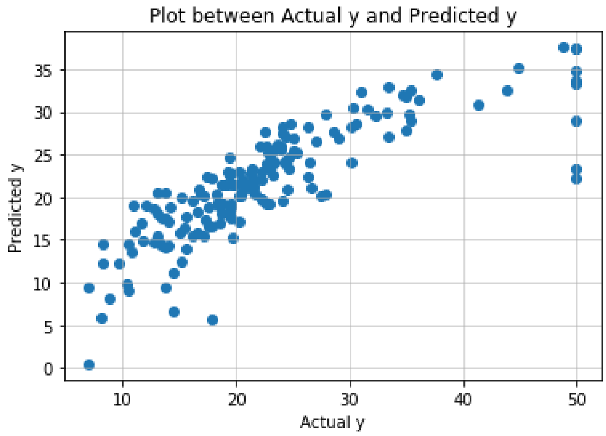
```
results.loc[0]=new
```

2. n_iter=100, lr_rate=0.01, lr_rate_variation='constant'

In [12]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='constant', eta0=0.01, n_iter=1
00)
```

```
C:\Users\kingsubham27091995\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)
```




```
mean sq error= 31.35249232605329
number of iteration= 100
Weight = [-0.85942537  0.72320993  0.15241913 -0.20932514 -1.58958236  2.10220312
 -0.42111051 -2.62106115  2.84313703 -2.11489664 -1.95183196  0.66487723
 -2.91904308]
```

In [13]:

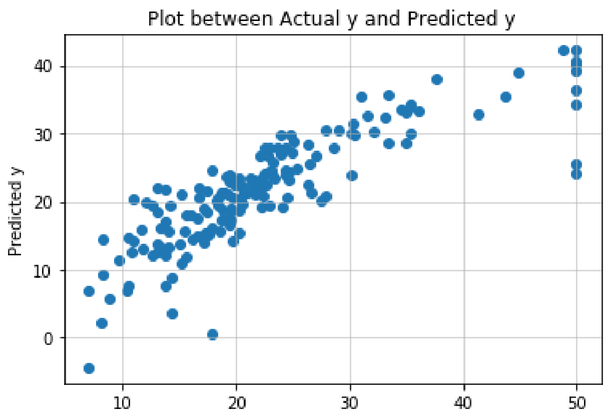
```
new=[3, 'SGDRegressor', 0.0001, 'constant', 0.01, 0.25, 100, error_sgd,w_sgd]
results.loc[1]=new
```

3. n_iter=1000, lr_rate=0.01, lr_rate_variation='constant'

In [14]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='constant', eta0=0.01, n_iter=1
000)
```

```
C:\Users\kingsubham27091995\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)
```



```
mean sq error= 27.446322247880957
number of iteration= 1000
Weight = [-0.84902213  0.72743498 -0.21789725  0.29233084 -1.48915036  2.91588396
 -0.14761927 -3.05222565  3.30506591 -2.10578481 -2.12863861  1.364138
 -3.49369362]
```

In [15]:

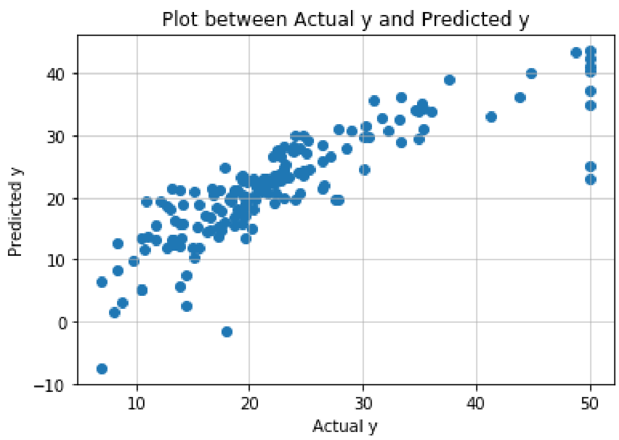
```
new=[5, 'SGDRegressor', 0.0001, 'constant', 0.01, 0.25, 1000, error_sgd,w_sgd]
results.loc[2]=new
```

4. n_iter=10000, lr_rate=0.01, lr_rate_variation='constant'

In [16]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='constant', eta0=0.01, n_iter=1
0000)
```

```
C:\Users\kingsubham27091995\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)
```




```
mean sq error= 27.99941966239229
number of iteration= 10000
Weight = [-1.13517669  0.86647558 -0.10205221  0.38522716 -1.21459202  3.35612688
 -0.17778189 -2.97663963  3.13116798 -2.18736619 -1.95867899  1.27876781
 -3.66442607]
```

In [17]:

```
new=[7, 'SGDRegressor', 0.0001, 'constant', 0.01, 0.25, 10000, error_sgd,w_sgd]
results.loc[3]=new
```

Manual SGD

In [47]:

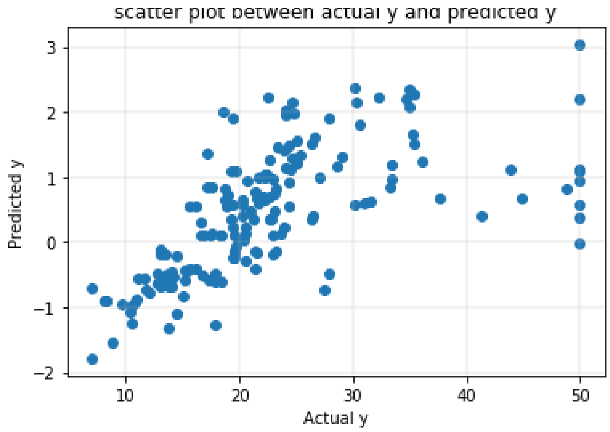
```
# this function is a simple implementation of sgd to linear regression, here we didn't use any reg
ularization
# we need to provide the pandas data with price, initial learning rate , and learning rate variati
on, number of iteration
# here we have implemented constant learning rate and invscaling learning rate
# checking the significant difference in loss i.e stopping condition might take lots of time so he
re we fix the number of loop
# this function returns weight (w) and bias (b)
# here we have taken sgd with batch size=10
def manual_fit(X, lr_rate_variation, alpha=0.0001, lr_rate=0.01, power_t=0.25, n_iter=100):
    w_new=np.zeros(shape=(1,13))
```

```
    w_new=np.zeros(shape=(1,13))
    b_new=0
    t=1
    r=lr_rate

    while(t<=n_iter):
        w_old=w_new
        b_old=b_new
        w_=np.zeros(shape=(1,13))
        b_=0
        x_data=X.sample(10)
        x=np.array(x_data.drop('price',axis=1))
        y=np.array(x_data['price'])


        for i in range(10):  # for getting the derivatives using sgd with k=10
            y_curr=np.dot(w_old,x[i])+b_old
            w_+=x[i] * (y[i] - y_curr)
            b_+=(y[i]-y_curr)


        w_*=(-2/x.shape[0])
        b_*=(-2/x.shape[0])

        #updating the parameters
        w_new=(w_old-r*w_)
        b_new=(b_old-r*b_)


        if(lr_rate_variation=='invscaling'):
            r = lr_rate / pow(t, power_t)
        t+=1



    return w_new, b_new



def pred(x,w, b):
    y_pred=[]
    for i in range(len(x)):
        y=np.asscalar(np.dot(w,x[i])+b)
        y_pred.append(y)
    return np.array(y_pred)


def plot_(X_test,y_pred):
    #scatter plot
    plt.scatter(y_test,y_pred)
    plt.grid(b=True, linewidth=0.3)
    plt.title('scatter plot between actual y and predicted y')
    plt.xlabel('Actual y')
    plt.ylabel('Predicted y')
    plt.show()



       ## Mean Squared Error
    mse_manual=mean_squared_error(y_test,y_pred)
    print('error=',mse_manual)


    return mse_manual
```

1. n_iter=1, lr_rate=0.01, lr_rate_variation='constant'

In [49]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='constant' , n_iter=1)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```




```
error= 571.8648648792249
Weights [[-0.11021193  0.19132733 -0.07735062  0.15245408 -0.09181451 -0.00957943
  -0.20430057  0.09798866 -0.04102483 -0.01402332 -0.11377034 -0.01153909
  -0.26066012]]
```

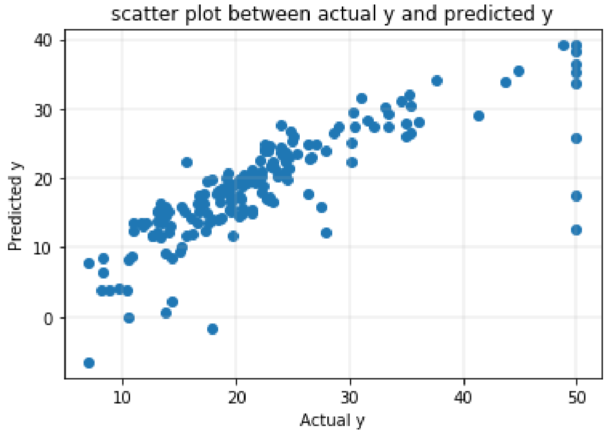
In [20]:

```
new=[2, 'manual sgd', 0.0001, 'constant', 0.01, 0.25, 1, manual_error,w]
results.loc[4]=new
```

2. n_iter=100, lr_rate=0.01, lr_rate_variation='constant'

In [50]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='constant' , n_iter=100)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```




```
error= 45.247347985147584
Weights [[-0.91092393 -0.39865859 -0.46309491  0.46539623 -0.33274583  3.75372959
  -0.59358478 -1.14425155 -0.12028126 -0.55272547 -1.97045747  0.03828791
  -2.48473031]]
```

In [23]:

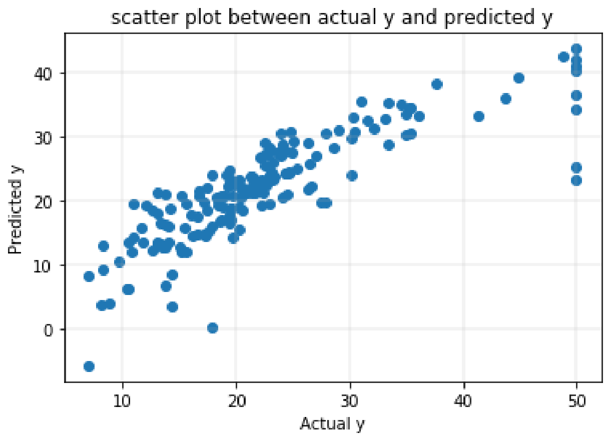
```
new=[4, 'manual sgd', 0.0001, 'constant', 0.01, 0.25, 100, manual_error,w]
results.loc[5]=new
```

3. n_iter=1000, lr_rate=0.01, lr_rate_variation='constant'

In [51]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='constant' , n_iter=1000)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```

```
print('Weights',w)
```




```
error= 27.237459972956998
Weights [[-1.09842893  0.71713151 -0.34491403  0.43390619 -1.51403988  3.07271161
  -0.30475742 -2.93154794  2.52490091 -1.23061076 -2.28491024  1.15251541
  -3.44014592]]
```

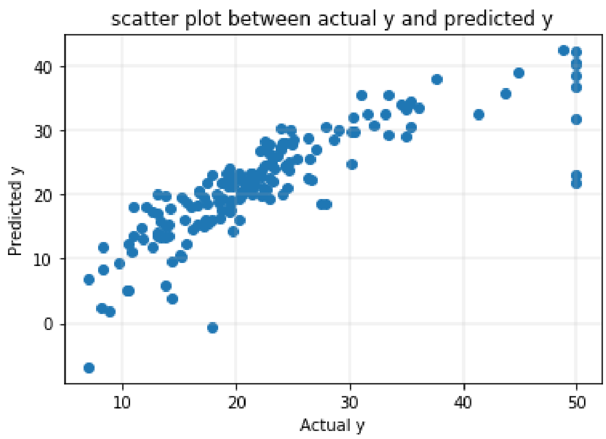
In [25]:

```
new=[6, 'manual sgd', 0.0001, 'constant', 0.01, 0.25, 1000, manual_error,w]
results.loc[6]=new
```

4. n_iter=10000, lr_rate=0.01, lr_rate_variation='constant'

In [52]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='constant' , n_iter=10000)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```




```
error= 28.911523488569138
Weights [[-1.2810705   0.62158022 -0.2314401   0.21230283 -1.29425501  3.02833457
  -0.28493935 -2.74803533  2.73872941 -2.29892697 -2.12457975  1.07042602
  -3.27696886]]
```

In [27]:

```
new=[8, 'manual sgd', 0.0001, 'constant', 0.01, 0.25, 10000, manual_error,w]
results.loc[7]=new
```

In [28]:

```
results
```

Out[28]:

	S_No	Algorithm	Alpha	LearningRateVariation	Learning Rate	Power_t	No. of Iterations	Mean Squared Error	Weights
0	1	SGDRegressor	0.0001	constant	0.01	0.25	1	28.850593	[-1.1565381407732438, 0.6697488843919033, -0.0...
1	3	SGDRegressor	0.0001	constant	0.01	0.25	100	31.352492	[-0.8594253660728869, 0.7232099282317112, 0.15...
2	5	SGDRegressor	0.0001	constant	0.01	0.25	1000	27.446322	[-0.8490221325906337, 0.7274349802523941, -0.2...
3	7	SGDRegressor	0.0001	constant	0.01	0.25	10000	27.999420	[-1.135176685101957, 0.8664755810330862, -0.10...
4	2	manual sgd	0.0001	constant	0.01	0.25	1	570.095994	[[-0.16385130745117604, 0.08744690043599387, -...
5	4	manual sgd	0.0001	constant	0.01	0.25	100	39.942861	[[-0.6219409715223574, 0.7468919969577941, -0....
6	6	manual sgd	0.0001	constant	0.01	0.25	1000	28.414085	[[-1.2674707278739434, 0.7273049238948522, -0....
7	8	manual sgd	0.0001	constant	0.01	0.25	10000	27.826577	[[-1.2047934560688556, 0.9175234573905571, -0....

Observation:

1. Fixed learning rate to 0.01 and lr_rate_variation='constant', changed n_iter
2. With increasing the number of iterations in ManualSGD,error is reducing.
3. With increase in iteration the number of element, ManualSGD weight and SGDRegressor weight seems similar

B) Using Learning Rate=0.01 , Learning rate variation='invscaling' and changing the n_iter

SGD Regressor

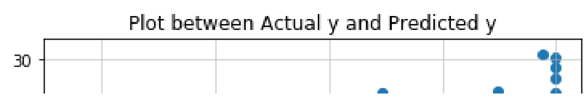
---

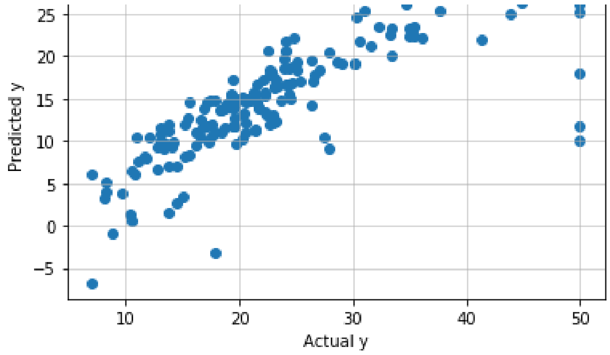
1. n_iter=1, lr_rate=0.01, lr_rate_variation='invscaling'

In [29]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='invscaling', eta0=0.01, n_iter
=1)
```

```
C:\Users\kingsubham27091995\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)
```






```
mean sq error= 102.99765448462009
number of iteration= 1
Weight = [-0.88705515  0.5677459  -0.51420203  0.03466216 -0.02705333  2.6051155
 -0.28325903 -1.11081536 -0.12236553 -0.45599899 -1.61426258  0.653454
 -1.75822116]
```

In [31]:

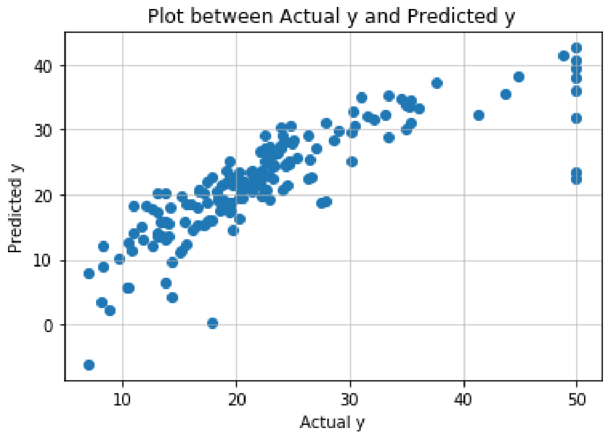
```
new=[9, 'SGDRegressor', 0.0001, 'invscaling', 0.01, 0.25, 1, error_sgd,w_sgd]
results.loc[8]=new
```

2. n_iter=100, lr_rate=0.01, lr_rate_variation='invscaling'

In [32]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='invscaling', eta0=0.01, n_iter
=100)
```

```
C:\Users\kingsubham27091995\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)
```




```
mean sq error= 28.449168811451475
number of iteration= 100
Weight = [-1.29336239  0.82854254 -0.27497766  0.20101891 -1.47344726  2.79825134
 -0.33875611 -2.79947307  2.61375872 -1.85723028 -2.12409713  1.0481751
 -3.31994023]
```

In [33]:

```
new=[11, 'SGDRegressor', 0.0001, 'invscaling', 0.01, 0.25, 100, error_sgd,w_sgd]
results.loc[9]=new
```

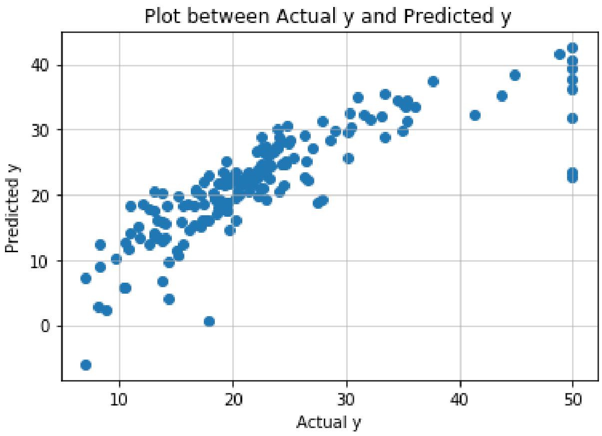
3. n_iter=1000, lr_rate=0.01, lr_rate_variation='invscaling'

In [34]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='invscaling', eta0=0.01, n_iter
```

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='invscaling', eta0=0.01, n_iter
=1000)
```

```
C:\Users\kingsubham27091995\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)
```




```
mean sq error= 28.4216616039839
number of iteration= 1000
Weight = [-1.30239854  0.85222639 -0.15526577  0.18210707 -1.47615506  2.77713541
 -0.31543693 -2.78215437  2.99057655 -2.25975291 -2.12295185  1.04981189
 -3.32165927]
```

In [35]:

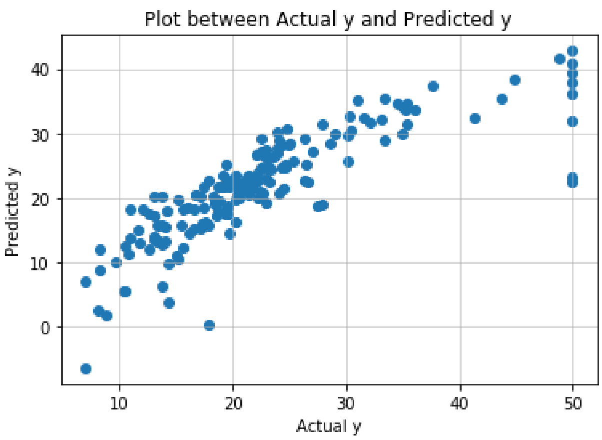
```
new=[13, 'SGDRegressor', 0.0001, 'invscaling', 0.01, 0.25, 1000, error_sgd,w_sgd]
results.loc[10]=new
```

4. n_iter=10000, lr_rate=0.01, lr_rate_variation='invscaling'

In [36]:

```
w_sgd, b_sgd, error_sgd=sklearn_sgd(alpha=0.0001, lr_rate_variation='invscaling', eta0=0.01, n_iter
=10000)
```

```
C:\Users\kingsubham27091995\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)
```




```
mean sq error= 28.59404884479356
number of iteration= 10000
Weight = [-1.31343832  0.86588662 -0.1722845   0.18760379 -1.49323433  2.79333583
 -0.32900798 -2.76801585  2.97317809 -2.27912365 -2.13716449  1.06037948
 -3.34155547]
```

In [37]:

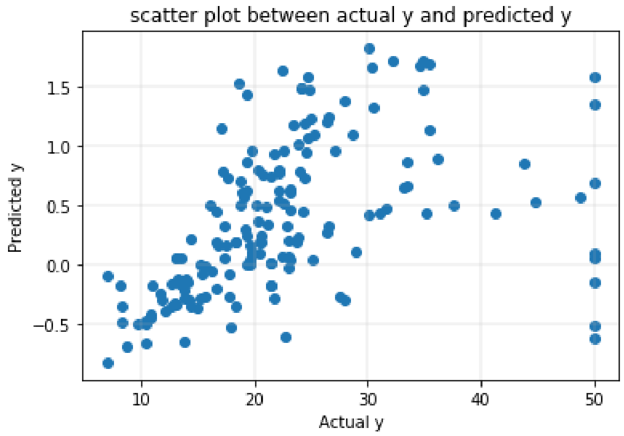
```
new=[15, 'SGDRegressor', 0.0001, 'invscaling', 0.01, 0.25, 10000, error_sgd,w_sgd]
results.loc[11]=new
```

Manual SGD

1. manual sgd, n_iter=1, lr_rate=0.01, lr_rate_variation='invscaling'

In [53]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='invscaling' , n_iter=1)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```




```
error= 579.7049068145751
Weights [[-0.05181692  0.11972694  0.04467098 -0.10294542 -0.10252764  0.02233647
  -0.16686319  0.11033126 -0.06713689 -0.02492568 -0.09296145 -0.02184467
  -0.09250086]]
```

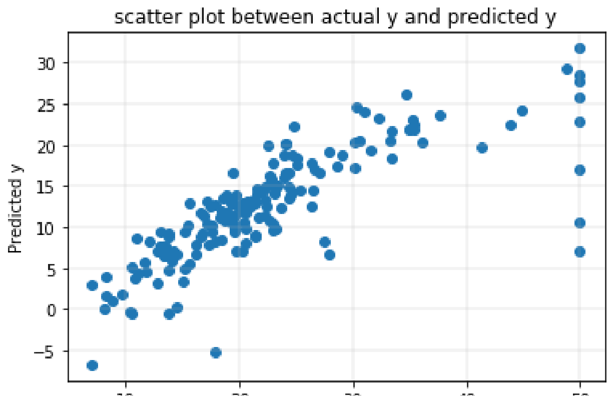
In [39]:

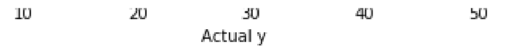
```
new=[10, 'manual sgd', 0.0001, 'invscaling', 0.01, 0.25, 1, manual_error,w]
results.loc[12]=new
```

2. n_iter=100, lr_rate=0.01, lr_rate_variation='invscaling'

In [54]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='invscaling' , n_iter=100)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```






```
error= 133.50034593501866
Weights [[-0.26878887  0.87360261 -0.57196422  0.45583587 -0.22374996  2.68056001
   0.23336127 -0.5546656  -0.06070142 -0.7084285  -1.79297507  0.84340591
  -1.60932098]]
```

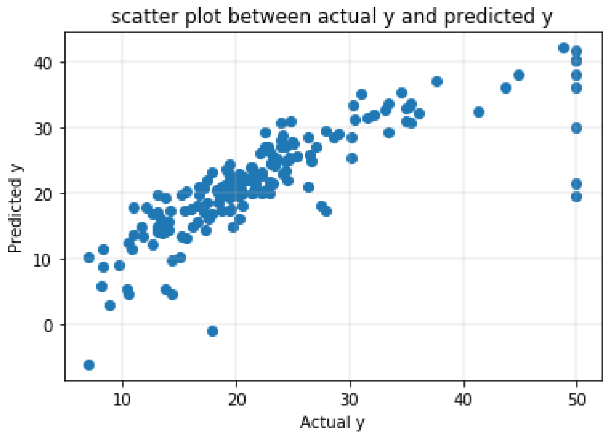
In [41]:

```
new=[12, 'manual sgd', 0.0001, 'invscaling', 0.01, 0.25, 100, manual_error,w]
results.loc[13]=new
```

3. n_iter=1000, lr_rate=0.01, lr_rate_variation='invscaling'

In [55]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='invscaling' , n_iter=1000)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```




```
error= 30.546060836678777
Weights [[-1.10803957  0.45079648 -0.52387316  0.13347052 -0.70026925  3.27626191
  -0.52264276 -2.01889623  0.85852156 -0.46609106 -2.07654475  0.98998175
  -2.95361238]]
```

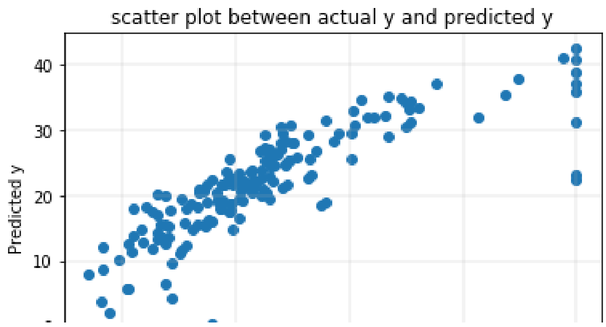
In [43]:

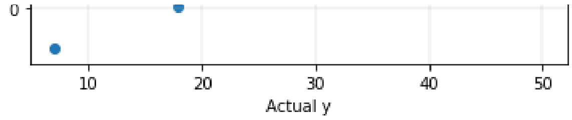
```
new=[14, 'manual sgd', 0.0001, 'invscaling', 0.01, 0.25, 1000, manual_error,w]
results.loc[14]=new
```

4. n_iter=10000, lr_rate=0.01, lr_rate_variation='invscaling'

In [56]:

```
w, b=manual_fit(X=manual_train, lr_rate_variation='invscaling' , n_iter=10000)
y_pred=pred(X_test, w=w, b=b)
manual_error=plot_(X_test,y_pred)
print('Weights',w)
```






```
error= 28.936981524682626
Weights [[-1.26070718  0.87652726 -0.386273    0.15368857 -1.44930645  2.71559288
  -0.39834965 -2.77354308  2.42358659 -1.72069068 -2.04099146  1.08625862
  -3.2961996 ]]
```

In [45]:

```
new=[16, 'manual sgd', 0.0001, 'invscaling', 0.01, 0.25, 10000, manual_error,w]
results.loc[15]=new
```

In [46]:

```
results
```

Out[46]:

	S_No	Algorithm	Alpha	LearningRateVariation	Learning Rate	Power_t	No. of Iterations	Mean Squared Error	Weights
0	1	SGDRegressor	0.0001	constant	0.01	0.25	1	28.850593	[-1.1565381407732438, 0.6697488843919033, -0.0...
1	3	SGDRegressor	0.0001	constant	0.01	0.25	100	31.352492	[-0.8594253660728869, 0.7232099282317112, 0.15...
2	5	SGDRegressor	0.0001	constant	0.01	0.25	1000	27.446322	[-0.8490221325906337, 0.7274349802523941, -0.2...
3	7	SGDRegressor	0.0001	constant	0.01	0.25	10000	27.999420	[-1.135176685101957, 0.8664755810330862, -0.10...
4	2	manual sgd	0.0001	constant	0.01	0.25	1	570.095994	[[-0.16385130745117604, 0.08744690043599387, -...
5	4	manual sgd	0.0001	constant	0.01	0.25	100	39.942861	[[-0.6219409715223574, 0.7468919969577941, -0....
6	6	manual sgd	0.0001	constant	0.01	0.25	1000	28.414085	[[-1.2674707278739434, 0.7273049238948522, -0....
7	8	manual sgd	0.0001	constant	0.01	0.25	10000	27.826577	[[-1.2047934560688556, 0.9175234573905571, -0....
8	9	SGDRegressor	0.0001	invscaling	0.01	0.25	1	102.997654	[-0.8870551536554298, 0.5677459020587332, -0.5...
9	11	SGDRegressor	0.0001	invscaling	0.01	0.25	100	28.449169	[-1.2933623861904127, 0.8285425375267194, -0.2...
10	13	SGDRegressor	0.0001	invscaling	0.01	0.25	1000	28.421662	[-1.302398535012179, 0.8522263897707502, -0.15...
11	15	SGDRegressor	0.0001	invscaling	0.01	0.25	10000	28.594049	[-1.3134383232601816, 0.8658866233270134, -

	S_No	Algorithm	Alpha	LearningRateVariation	Learning Rate	Power_t	No. of Iterations	Mean Squared Error	Weights
12	10	manual sgd	0.0001	invscaling	0.01	0.25	1	576.576234	[[-0.15742251779771546, 0.050568062836999064, ...
13	12	manual sgd	0.0001	invscaling	0.01	0.25	100	137.622755	[[-0.6542615204524277, 0.34454609527638236, -0...
14	14	manual sgd	0.0001	invscaling	0.01	0.25	1000	29.198173	[[-0.9535008837079129, 0.5690156703435716, -0....
15	16	manual sgd	0.0001	invscaling	0.01	0.25	10000	28.536891	[[-1.2540957654038705, 0.8617983337484499, -0....

In [59]:

```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable

x = PrettyTable()
x.field_names = ["Algorithm", "LearningRateVariation", "Weights"]

x.add_row(["SGDRegressor", "constant",'''-1.15653814  0.66974888 -0.08178764  0.6382138  -
0.66797691  2.95215349
 -0.10357986 -1.93735889  1.12825615 -0.2267697  -1.78106711  0.95286871
 -2.90881683''' ])
x.add_row(["SGDRegressor", "constant",'''-0.85942537  0.72320993  0.15241913 -0.20932514 -
1.58958236  2.10220312
 -0.42111051 -2.62106115  2.84313703 -2.11489664 -1.95183196  0.66487723
 -2.91904308''' ])
x.add_row(["SGDRegressor", "constant",'''-0.84902213  0.72743498 -0.21789725  0.29233084 -
1.48915036  2.91588396
 -0.14761927 -3.05222565  3.30506591 -2.10578481 -2.12863861  1.364138
 -3.49369362''' ])
x.add_row(["SGDRegressor", "constant",'''-1.13517669  0.86647558 -0.10205221  0.38522716 -
1.21459202  3.35612688
 -0.17778189 -2.97663963  3.13116798 -2.18736619 -1.95867899  1.27876781
 -3.66442607''' ])
x.add_row(["Manual SGD", "constant",'''-0.13911115  0.0308758  -0.2862618  -0.13815142 -0.01849442
0.44716359
   0.02252312  0.06645053 -0.08661799 -0.18113118 -0.19971864  0.18560991
  -0.38848135''' ])
x.add_row(["Manual SGD", "constant",'''-0.91092393 -0.39865859 -0.46309491  0.46539623 -0.33274583
3.75372959
  -0.59358478 -1.14425155 -0.12028126 -0.55272547 -1.97045747  0.03828791
  -2.48473031''' ])
x.add_row(["Manual SGD", "constant", '''-1.09842893  0.71713151 -0.34491403  0.43390619 -
1.51403988  3.07271161
  -0.30475742 -2.93154794  2.52490091 -1.23061076 -2.28491024  1.15251541
  -3.44014592'''])
x.add_row(["Manual SGD", "constant", '''-1.2810705   0.62158022 -0.2314401   0.21230283 -
1.29425501  3.02833457
  -0.28493935 -2.74803533  2.73872941 -2.29892697 -2.12457975  1.07042602
  -3.27696886'''])
x.add_row(["SGDRegressor", "invscaling",'''-0.88705515  0.5677459  -0.51420203  0.03466216 -
0.02705333  2.6051155
 -0.28325903 -1.11081536 -0.12236553 -0.45599899 -1.61426258  0.653454
 -1.75822116''' ])
x.add_row(["SGDRegressor", "invscaling",'''-1.29336239  0.82854254 -0.27497766  0.20101891 -
1.47344726  2.79825134
 -0.33875611 -2.79947307  2.61375872 -1.85723028 -2.12409713  1.0481751
 -3.31994023''' ])
x.add_row(["SGDRegressor", "invscaling", '''-1.30239854  0.85222639 -0.15526577  0.18210707 -
1.47615506  2.77713541
 -0.31543693 -2.78215437  2.99057655 -2.25975291 -2.12295185  1.04981189
 -3.32165927'''])
x.add_row(["SGDRegressor", "invscaling",'''-1.31343832  0.86588662 -0.1722845   0.18760379 -
```

```
1.49323433  2.79333583
 -0.32900798 -2.76801585  2.97317809 -2.27912365 -2.13716449  1.06037948
 -3.34155547 ''']) 
x.add_row(["Manual SGD", "invscaling",'''-0.05181692  0.11972694  0.04467098 -0.10294542 -
0.10252764  0.02233647
  -0.16686319  0.11033126 -0.06713689 -0.02492568 -0.09296145 -0.02184467
  -0.09250086''' ]) 
x.add_row(["Manual SGD", "invscaling",'''-0.26878887  0.87360261 -0.57196422  0.45583587 -
0.22374996  2.68056001
   0.23336127 -0.5546656  -0.06070142 -0.7084285  -1.79297507  0.84340591
  -1.60932098''' ]) 
x.add_row(["Manual SGD", "invscaling",'''-1.10803957  0.45079648 -0.52387316  0.13347052 -
0.70026925  3.27626191
  -0.52264276 -2.01889623  0.85852156 -0.46609106 -2.07654475  0.98998175
  -2.95361238 ''']) 
x.add_row(["Manual SGD", "invscaling",'''-1.26070718  0.87652726 -0.386273    0.15368857 -
1.44930645  2.71559288
  -0.39834965 -2.77354308  2.42358659 -1.72069068 -2.04099146  1.08625862
  -3.2961996 ''']) 

print(x)
```

```
+--------------+-----------------------+-------------------------------------------------------------
------------+
|  Algorithm   | LearningRateVariation |                                 Weights
|
+--------------+-----------------------+-------------------------------------------------------------
------------+
| SGDRegressor |        constant       |  -1.15653814  0.66974888 -0.08178764  0.6382138  -0.667976
91  2.95215349  |
|              |                       |   -0.10357986 -1.93735889  1.12825615 -0.2267697  -1.78106
11  0.95286871 |
|              |                       |                                -2.90881683
|
| SGDRegressor |        constant       |  -0.85942537  0.72320993  0.15241913 -0.20932514 -1.589582
36  2.10220312  |
|              |                       |   -0.42111051 -2.62106115  2.84313703 -2.11489664 -1.95183
96  0.66487723 |
|              |                       |                                -2.91904308
|
| SGDRegressor |        constant       |  -0.84902213  0.72743498 -0.21789725  0.29233084 -1.489150
36  2.91588396  |
|              |                       |    -0.14761927 -3.05222565  3.30506591 -2.10578481 -2.1286
861  1.364138   |
|              |                       |                                -3.49369362
|
| SGDRegressor |        constant       |  -1.13517669  0.86647558 -0.10205221  0.38522716 -1.214592
02  3.35612688  |
|              |                       |   -0.17778189 -2.97663963  3.13116798 -2.18736619 -1.95867
99  1.27876781 |
|              |                       |                                -3.66442607
|
|  Manual SGD  |        constant       |  -0.13911115  0.0308758  -0.2862618  -0.13815142 -0.018494
42  0.44716359  |
|              |                       |    0.02252312  0.06645053 -0.08661799 -0.18113118 -0.19971
64  0.18560991 |
|              |                       |                                -0.38848135
|
|  Manual SGD  |        constant       |  -0.91092393 -0.39865859 -0.46309491  0.46539623 -0.332745
83  3.75372959  |
|              |                       |   -0.59358478 -1.14425155 -0.12028126 -0.55272547 -1.97045
47  0.03828791 |
|              |                       |                                -2.48473031
|
|  Manual SGD  |        constant       |  -1.09842893  0.71713151 -0.34491403  0.43390619 -1.514039
88  3.07271161  |
|              |                       |   -0.30475742 -2.93154794  2.52490091 -1.23061076 -2.28491
24  1.15251541 |
|              |                       |                                -3.44014592
|
|  Manual SGD  |        constant       |  -1.2810705   0.62158022 -0.2314401   0.21230283 -1.294255
01  3.02833457  |
|              |                       |   -0.28493935 -2.74803533  2.73872941 -2.29892697 -2.12457
75  1.07042602 |
|              |                       |                                -3.27696886
|
```

```
| SGDRegressor |       invscaling      |   -0.88705515  0.5677459  -0.51420203  0.03466216 -0.02705
333  2.6051155  |
|              |                       |    -0.28325903 -1.11081536 -0.12236553 -0.45599899 -1.6142
258  0.653454  |
|              |                       |                                 -1.75822116
|
| SGDRegressor |       invscaling      |  -1.29336239  0.82854254 -0.27497766  0.20101891 -1.473447
26  2.79825134  |
|              |                       |   -0.33875611 -2.79947307  2.61375872 -1.85723028 -2.12409
13  1.0481751  |
|              |                       |                                 -3.31994023
|
| SGDRegressor |       invscaling      |  -1.30239854  0.85222639 -0.15526577  0.18210707 -1.476155
06  2.77713541  |
|              |                       |   -0.31543693 -2.78215437  2.99057655 -2.25975291 -2.12295
85  1.04981189 |
|              |                       |                                 -3.32165927
|
| SGDRegressor |       invscaling      |  -1.31343832  0.86588662 -0.1722845   0.18760379 -1.493234
33  2.79333583  |
|              |                       |   -0.32900798 -2.76801585  2.97317809 -2.27912365 -2.13716
49  1.06037948 |
|              |                       |                                 -3.34155547
|
|  Manual SGD  |       invscaling      |  -0.05181692  0.11972694  0.04467098 -0.10294542 -0.102527
64  0.02233647  |
|              |                       |   -0.16686319  0.11033126 -0.06713689 -0.02492568 -0.09296
45 -0.02184467 |
|              |                       |                                 -0.09250086
|
|  Manual SGD  |       invscaling      |  -0.26878887  0.87360261 -0.57196422  0.45583587 -0.22374
96  2.68056001  |
|              |                       |    0.23336127 -0.5546656  -0.06070142 -0.7084285  -1.79297
07  0.84340591 |
|              |                       |                                 -1.60932098
|
|  Manual SGD  |       invscaling      |  -1.10803957  0.45079648 -0.52387316  0.13347052 -0.70026
25  3.27626191  |
|              |                       |   -0.52264276 -2.01889623  0.85852156 -0.46609106 -2.07654
75  0.98998175 |
|              |                       |                                 -2.95361238
|
|  Manual SGD  |       invscaling      |  -1.26070718  0.87652726 -0.386273    0.15368857 -1.449306
45  2.71559288  |
|              |                       |   -0.39834965 -2.77354308  2.42358659 -1.72069068 -2.04099
46  1.08625862 |
|              |                       |                                 -3.2961996
|
+--------------+-----------------------+-------------------------------------------------------------
------------+
```

Conclusion:

- In both 'constant' and 'invscaling' implementation in SGDRegressor and ManualSGD Regressor, with higher number of iteration , manual sgd seems similar to SGDRegressor
- In the results dataframe we can see that , In Manual SGD , error reduces with increasing in iteration number