```
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
```

```
X = load_boston().data
Y = load_boston().target
df=pd.DataFrame(X)
#some intuition
df[13]=df[10]//df[12]  #here we set a column 13 such that df[13]=Boston_data['Medv']//Boston_data['B
X=df.as_matrix()
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	3.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	1.0
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	4.0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	6.0
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	3.0

```
#Splitting whole data into train and test
from sklearn.model_selection  import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X, Y, test_size=0.3, random_state=4)

# applying column standardization on train and test data

scaler = preprocessing.StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)

data_train=pd.DataFrame(X_train)
data_train['price']=y_train
data_train.head()
```

	0	1	2	3	4	5	6	7	
0	-0.425469	-0.470768	-0.954686	-0.231455	-0.919581	0.215100	-0.747410	0.454022	-0.7
1	-0.426323	2.992576	-1.330157	-0.231455	-1.227311	-0.883652	-1.691588	3.163428	-0.6

```
#SGD implementation for linear regression

W,B,iteration,lr_rate,k=np.zeros(shape=(1,14)),0,750,0.01,25 #intialise W and B to zero

while iteration>=0:
    w,b,temp_vectors,temp_intercept=W,B,np.zeros(shape=(1,14)),0
    data=data_train.sample(25) #sampling random k=batch size=20 data
    x=np.array(data.drop('price',axis=1))
    y=np.array(data['price'])

    for i in range(k):
        temp_vectors+=(-2)*x[i]*(y[i]-(np.dot(w,x[i])+b))#partial differentiation wrt w dl/dw=1/k(-2
        temp_intercept+=(-2)*(y[i]-(np.dot(w,x[i])+b))#partial differentiation wrt b dl/db=1/k(-2)*(

    W=(w-lr_rate*(temp_vectors)/k)
    B=(b-lr_rate*(temp_intercept)/k)

    iteration-=1


print(W)
print(B)
```

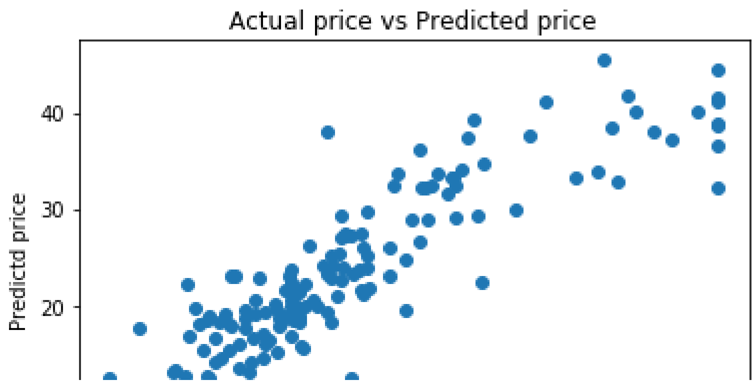
```
[[-1.07154866  0.6824004  -0.73812727  0.94985895 -1.53072618  1.83740877
   0.12717258 -3.04013226  1.95119164 -1.05883575 -2.31322222  0.84870653
  -1.83316004  2.73721903]]
[22.13209163]
```

```
#prediction on x_test

y_predic_linear_regression = [ ]
for i in range(len(X_test)):
    val=np.dot(W,X_test[i])+B  #val= wTx+b
    y_predic_linear_regression.append(np.asscalar(val))


#Scatter plot of actual price vs predicted price

plt.scatter(y_test, y_predic_linear_regression)
plt.xlabel('Actual price')
plt.ylabel('Predictd price')
plt.title('Actual price vs Predicted price')
plt.show()
```




```
MSE_linear_regression = mean_squared_error(y_test,y_predic_linear_regression)
print('mean squared error =',MSE_linear_regression)
```

```
mean squared error = 23.05958235093564
```

```
#SGD regression sklearn implementation

model = SGDRegressor(learning_rate ='constant',eta0 = 0.01, penalty = None, n_iter_no_change = 100,
model.fit(X_train,y_train)
y_pred_sgd = model.predict(X_test)

#Scatter plot of actual price vs predicted price

plt.scatter(y_test,y_pred_sgd)
plt.xlabel('Actual price')
plt.ylabel('Predictd price')
plt.title('Actual price vs Predicted price')
plt.show()
```


```
MSE_sgd=mean_squared_error(y_test,y_pred_sgd)
print('mean squared error =',MSE_sgd)
```

```
mean squared error = 25.215752448802053
```

```
#comparison between MSE of own implementation and SGD sklearn implementation
print('MSE of manual implementation = ',MSE_linear_regression)
print('='*50)
```

```
print('MSE of SGD sklearn implementation = ',MSE_sgd)
```

```
MSE of manual implementation =  23.05958235093564
==================================================
MSE of SGD sklearn implementation =  25.215752448802053
```