**SGD - Stochastic Gradient Descent**

**Link :-**

<https://docs.google.com/spreadsheets/d/1uL7t2QqQwdBq0KLugIu0m2luaj8ElZTF3z7CgCrRvJk/edit?usp=sharing>

**Demonstration code :-**

**import warnings**

**warnings.filterwarnings("ignore")**

**from sklearn.datasets import load\_boston**

**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**

**from numpy import random**

**from sklearn.model\_selection import train\_test\_split**

**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)**

**print("X Shape: ",X.shape)**

**print("Y Shape: ",Y.shape)**

**print("X\_Train Shape: ",x\_train.shape)**

**print("X\_Test Shape: ",x\_test.shape)**

**print("Y\_Train Shape: ",y\_train.shape)**

**print("Y\_Test Shape: ",y\_test.shape)**

**# standardizing data**

**scaler = preprocessing.StandardScaler().fit(x\_train)**

**x\_train = scaler.transform(x\_train)**

**x\_test=scaler.transform(x\_test)**

**## Adding the PRIZE Column in the data**

**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)**

**n\_iter=100**

**clf\_ = SGDRegressor(max\_iter=n\_iter)**

**clf\_.fit(x\_train, y\_train)**

**y\_pred\_sksgd=clf\_.predict(x\_test)**

**plt.scatter(y\_test,y\_pred\_sksgd)**

**plt.grid()**

**plt.xlabel('Actual y')**

**plt.ylabel('Predicted y')**

**plt.title('Scatter plot from actual y and predicted y')**

**plt.show()**

**print('Mean Squared Error :',mean\_squared\_error(y\_test, y\_pred\_sksgd))**

**def MyCustomSGD(train\_data,learning\_rate,n\_iter,k,divideby):**

**# Initially we will keep our W and B as 0 as per the Training Data**

**w=np.zeros(shape=(1,train\_data.shape[1]-1))**

**b=0**

**cur\_iter=1**

**while(cur\_iter<=n\_iter):**

**# We will create a small training data set of size K**

**temp=train\_data.sample(k)**

**# We create our X and Y from the above temp dataset**

**y=np.array(temp['price'])**

**x=np.array(temp.drop('price',axis=1))**

**# We keep our initial gradients as 0**

**w\_gradient=np.zeros(shape=(1,train\_data.shape[1]-1))**

**b\_gradient=0**

**for i in range(k): # Calculating gradients for point in our K sized dataset**

**prediction=np.dot(w,x[i])+b**

**w\_gradient=w\_gradient+(-2)\*x[i]\*(y[i]-(prediction))**

**b\_gradient=b\_gradient+(-2)\*(y[i]-(prediction))**

**#Updating the weights(W) and Bias(b) with the above calculated Gradients**

**w=w-learning\_rate\*(w\_gradient/k)**

**b=b-learning\_rate\*(b\_gradient/k)**

**# Incrementing the iteration value**

**cur\_iter=cur\_iter+1**

**#Dividing the learning rate by the specified value**

**learning\_rate=learning\_rate/divideby**

**return w,b #Returning the weights and Bias**

**def predict(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)**

**w,b=MyCustomSGD(train\_data,learning\_rate=1,n\_iter=100,divideby=2,k=10)**

**y\_pred\_customsgd=predict(x\_test,w,b)**

**plt.scatter(y\_test,y\_pred\_customsgd)**

**plt.grid()**

**plt.xlabel('Actual y')**

**plt.ylabel('Predicted y')**

**plt.title('Scatter plot from actual y and predicted y')**

**plt.show()**

**print('Mean Squared Error :',mean\_squared\_error(y\_test, y\_pred\_customsgd))**

**w,b=MyCustomSGD(train\_data,learning\_rate=0.001,n\_iter=1000,divideby=1,k=10)**

**y\_pred\_customsgd\_improved=predict(x\_test,w,b)**

**plt.scatter(y\_test,y\_pred\_customsgd\_improved)**

**plt.grid()**

**plt.xlabel('Actual y')**

**plt.ylabel('Predicted y')**

**plt.title('Scatter plot from actual y and predicted y')**

**plt.show()**

**print('Mean Squared Error :',mean\_squared\_error(y\_test, y\_pred\_customsgd\_improved))**

**Screenshots :-**

**(Images may take a while to load)**













