Name:- Krishna Mundada

Roll No:- 45

Batch:- E3

Write a program to implement

1. Vanilla gradient descent.
2. Momentum based gradient descent.

Consider two points - (2, 0.3), (6, 0.8) approximate these two points by using above mention algorithm, also plot approx. curve.
Learning Rate (lr) = 0.5
Wt+1 = Wt - lr * dWt

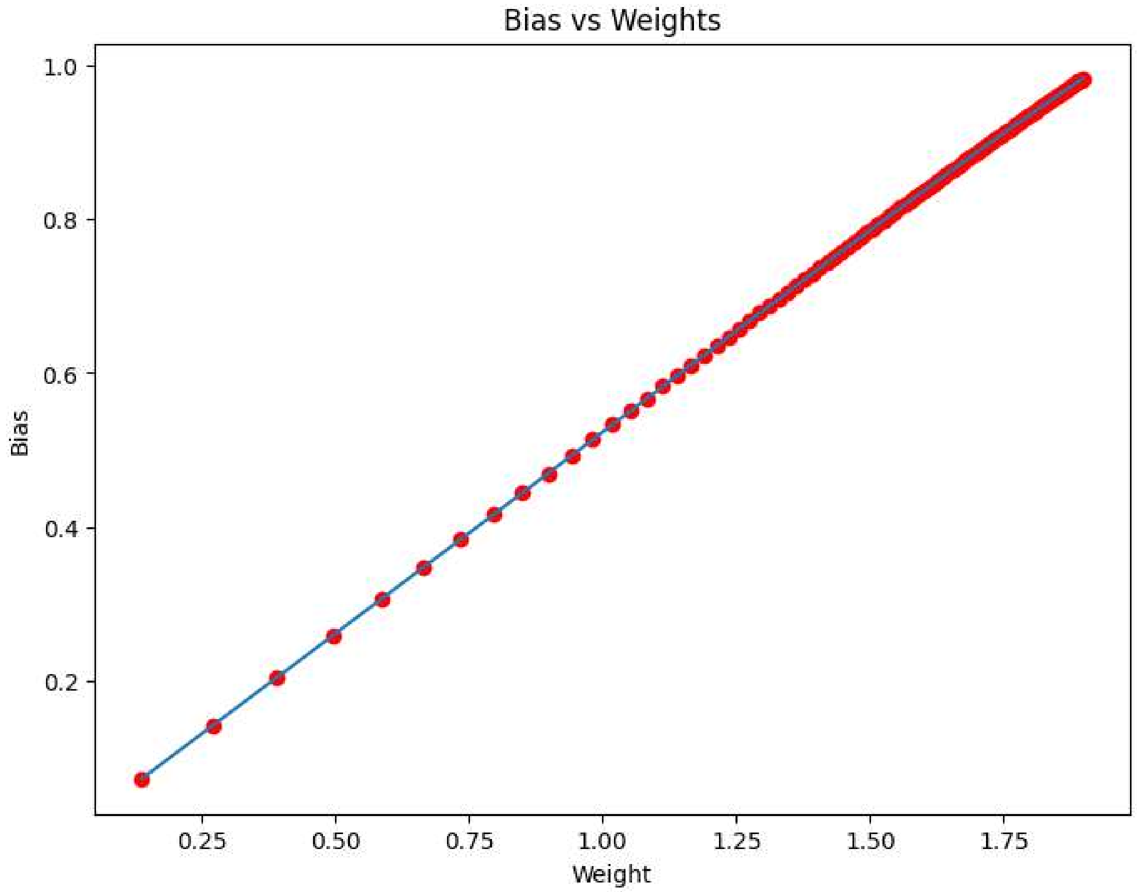
Vanilla Gradient Descent

In [ ]:
```
# Two points - (2, 0.3), (6, 0.8)
import numpy as np
X = [2,0.3] # point 1
Y = [6,0.8] # point 2

# Activation Function: Sigmoid
def f(x,w,b):
    return 1/(1+np.exp(-(w*x+b)))

def grad_b(x,w,b,y):
    fx = f(x,w,b)
    return (fx-y)*fx*(1-fx)

def grad_w(x,w,b,y):
    fx = f(x,w,b)
    return (fx-y)*fx*(1-fx)*x

i = 1
list_w = []
list_b = []
def vanila_gradient_descent():
    w,b,eta,max_epochs = 0,0,0.05,100
    for i in range(max_epochs):
        dw,db = 0,0
        for x,y in zip(X,Y):
            dw += grad_w(x,w,b,y)
            db += grad_b(x,w,b,y)
        w = w - eta*dw
        b = b - eta*db

        list_w.append(w)
        list_b.append(b)

        print('Weight: ', w, '     Bias: ', b)
```

```
        #print('Bias: ', b)
        print('Iteration ', i+1)
```

In [ ]:
```
vanila_gradient_descent()
```

In [ ]:
```
list_w
list_b

import matplotlib.pyplot as plt
plt.figure(figsize = (8,6))
plt.plot(list_w, list_b)
plt.scatter(list_w, list_b, marker='o', color='red')
plt.title("Bias vs Weights")
plt.ylabel("Bias")
plt.xlabel("Weight")
plt.show()
```



Momentum Based Gradient Descent

In [ ]:
```
x = [3, 2.5, 7]
y = [5, 4.5, 5]

learning_rate = 0.01
n_iterations = 200

import matplotlib.pyplot as plt
import numpy as np
import random
li = [0,1,2]
inter = list()
sl = list()
```

```
def sum_of_squared_intercept(intercept,slope,x,y):
    cost_intercept = -2*(y-(intercept + (slope*x)))
    return cost_intercept

def sum_of_square_slope(intercept,slope,x,y):
    cost_slope= -2*y*(y-(intercept + (slope*x)))
    return cost_slope
def momentum_gardient_descent():
    intercept = 0
    slope = 0
    j = random.choice(li)
    for i in range(200):
        print('Intercept: ' , intercept,' Slope: ',slope)
        inter.append(intercept)
        print('Iteration: ', i)
        sl.append(slope)
        cost_int = sum_of_squared_intercept(intercept,slope,x[j],y[j])
        cost_sl = sum_of_square_slope(intercept,slope,x[j],y[j])
        step_size_intercept = cost_int*learning_rate
        step_size_slope = cost_sl*learning_rate
        intercept=intercept-(step_size_intercept)
        slope=slope-step_size_slope

momentum_gardient_descent()
```

In [ ]:

```
import matplotlib.pyplot as plt
plt.figure(figsize = (8,6))
plt.plot(inter, sl)
plt.scatter(inter, sl, marker='o', color='black')
plt.show()
```