```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Create the DataFrame for train the model:

1. create random data using arrays for age, then classinfing it to child (0) or teenage (1).

```
#random ages
child = np.random.randint(1,11, 25)
tens = np.random.randint(13, 16, 25)

ages = np.concatenate((child, tens))

labels = []
#create labels for ages classification
for age in ages:
    if(age > 0 and age < 11):
        labels.append(0)
    elif(age > 12 and age < 16):
        labels.append(1)

#check from length:
print(f'Labels{len(labels)}, Ages{len(ages)}')
Labels50, Ages50</pre>
```

1. create the DataFrame that hold the arrays.

```
#create a DataFrame:
df = pd.DataFrame({'ages' : ages, 'labels' : labels})
df
    ages labels
0
       1
                0
1
       8
                0
2
       8
                0
3
       8
                0
4
       8
                0
5
       7
                0
6
       6
                0
7
       5
                0
       7
8
                0
9
       1
                0
10
       8
                0
       8
                0
11
12
       6
                0
13
       7
                0
14
       3
                0
```

```
15
        1
                  0
16
        1
                  0
        8
17
                  0
18
        6
                  0
        7
                  0
19
20
        9
                  0
        4
                  0
21
22
        4
                  0
23
        5
                  0
        9
                  0
24
25
       14
                  1
       13
                  1
26
27
       15
                  1
28
                  1
       14
29
       13
                  1
30
                  1
       15
31
       15
                  1
32
       13
                  1
33
       13
                  1
34
       15
                  1
35
       14
                  1
36
       15
                  1
37
                  1
       13
38
       13
                  1
39
       14
                  1
40
       14
                  1
41
       15
                  1
42
       13
                  1
43
       14
                  1
44
       13
                  1
45
       13
                  1
46
       15
                  1
47
       13
                  1
48
       15
                  1
49
       13
                  1
```

Write the code for Logistic algorithm:

```
#values for training:
x = df['ages']
y = df['labels']

#values for the model:
epoch = 1500
learn_rate = 0.1
w = 0
b = 0
losses = []
```

```
#function to calculate the result:
def loss fun(y true, y pred):
    #to sure the range of output between [0,1]:
    eps = 1e-15
    y_pred = np.clip(y_pred, eps, 1 - eps) #array, minimum value,
maximum value
    #loss fun : (1/n) ( -y-true log(y pred) - (1 - y true) log(y pred)
    result = y true * np.log(y pred) + (1 - y true) * np.log(1 - y true)
y_pred)
    return (-np.mean(result))
#create the main model:
for i in range(epoch):
    z = x*w + b
    sigmuid = 1. / (1. + np.exp(-np.clip(z, -250, 250)))
    error = sigmuid - y
    loss = loss fun(y, sigmuid)
    losses.append(loss)
    #evaluate drivatives:
    dw = (1 / len(x)) * np.dot(x,error)
    db = (1 / len(x)) * np.sum(error)
    #update values:
    w -= learn rate*dw
    b -= learn rate*db
#final results:
z = w*x + b
r = 1. / (1. + np.exp(-np.clip(z, -250, 250)))
predicted = np.where(r \ge 0.5, 1, 0)
# results:
z_final_result = w*x + b
print(f'Real values: ',y)
print(f'Predicted values:', predicted)
print(f'Loss: ', np.round(losses[-1], 2))
print(f'weight: ', np.round(w, 2))
print(f'intercept: ',np.round(b, 2))
```

Real values: 0 0 0 1 0 2 0 0 3 0 0 4 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0						
1	Real	values:	0	0		
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	1	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	2	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	3	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	4	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	5	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	6	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	7	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	8	0				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	9	Ö				
11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 25 1 26 1 27 1 28 1 29 1 30 1 31 1 31 1 32 1 33 1	10	0				
31 1 32 1 33 1 34 1	11	0				
31 1 32 1 33 1 34 1	12	0				
31 1 32 1 33 1 34 1	13	0				
31 1 32 1 33 1 34 1	14	0				
31 1 32 1 33 1 34 1	15	0				
31 1 32 1 33 1 34 1	16	0				
31 1 32 1 33 1 34 1	17	Õ				
31 1 32 1 33 1 34 1	18	0				
31 1 32 1 33 1 34 1	19	0				
31 1 32 1 33 1 34 1	20	Ö				
31 1 32 1 33 1 34 1	21	Ö				
31 1 32 1 33 1 34 1	22	Ö				
31 1 32 1 33 1 34 1	23	Ö				
31 1 32 1 33 1 34 1	24	Ö				
31 1 32 1 33 1 34 1	25	1				
31 1 32 1 33 1 34 1	26	1				
31 1 32 1 33 1 34 1	27	1				
31 1 32 1 33 1 34 1	28	1				
31 1 32 1 33 1 34 1	29	1				
31 1 32 1 33 1 34 1	30	1				
32 1 33 1 34 1	31	1				
34 1	32					
34 1	33	1				
	34	1				
36 1 37 1 38 1 39 1 40 1 41 1 42 1 43 1 44 1 45 1 46 1 47 1 48 1 49 1	35					
37 1 38 1 39 1 40 1 41 1 42 1 43 1 44 1 45 1 46 1 47 1 48 1 49 1	36	1				
38	37	1				
39	38	1				
40 1 41 1 42 1 43 1 44 1 45 1 46 1 47 1 48 1 49 1	39	1				
41 1 42 1 43 1 44 1 45 1 46 1 47 1 48 1 49 1	40	1				
42 1 43 1 44 1 45 1 46 1 47 1 48 1 49 1	41	1				
43 1 44 1 45 1 46 1 47 1 48 1 49 1	42	1				
44 1 45 1 46 1 47 1 48 1 49 1	43	1				
45 1 46 1 47 1 48 1 49 1	44	1				
46 1 47 1 48 1 49 1	45	1				
47 1 48 1 49 1	46	1				
48 1 49 1	47	1				
49 1	48	1				
	49	1				

Draw the chart for the loss of Logistic algorithm:

```
#draw a chart for loss:

plt.plot(np.arange(0, len(losses),1), np.array(losses))
plt.title('loss for logistic algorithm'.title())
plt.yticks([0, 0.5, 1])
plt.xlabel('Loss')
plt.show()
```

