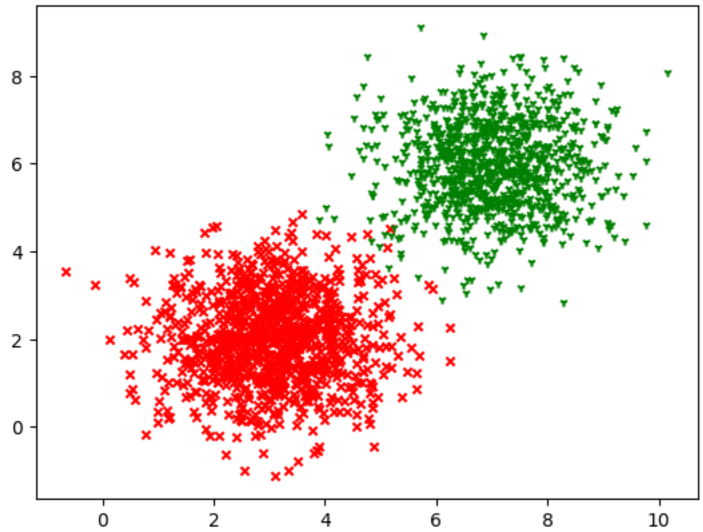
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np


import keras
from keras.models import Sequential
from keras.layers import Dense
```

Zbiór danych:

```
x_label1 = np.random.normal(3, 1, 1000)
y_label1 = np.random.normal(2, 1, 1000)
x_label2 = np.random.normal(7, 1, 1000)
y_label2 = np.random.normal(6, 1, 1000)

xs = np.append(x_label1, x_label2)
ys = np.append(y_label1, y_label2)
labels = np.asarray([0.]*len(x_label1)+[1.]*len(x_label2))


plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.show()
```



Funkcja błędu (entropia krzyżowa):

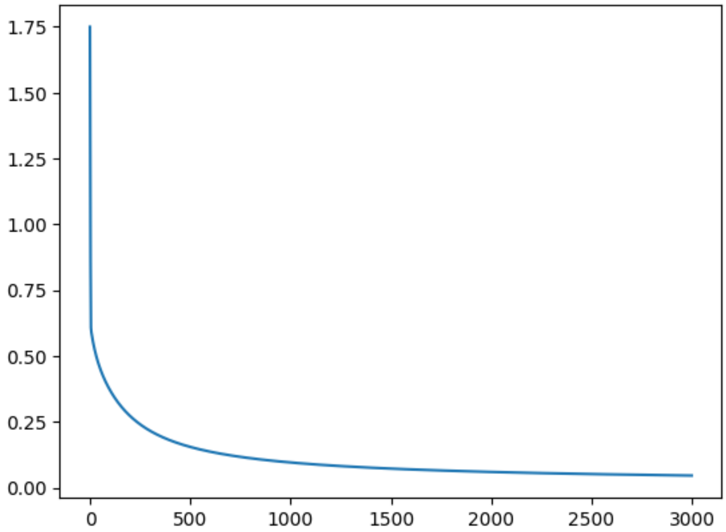
```
def loss_fn(label, label_model):
    return tf.reduce_mean(-label*tf.math.log(label_model)-(1-label)*tf.math.log(1-label_model))
```

Początkowe wartości parametrów i pętla ucząca:

```
import random
a = tf.Variable(random.random())
b = tf.Variable(random.random())

c = tf.Variable(random.random())

Loss = []
epochs = 3000
lr = 0.1

for _ in range(epochs):

  with tf.GradientTape() as tape:
    #predykcja modelu

    pred_y = tf.sigmoid(a * xs + b * ys + c)
    #policzenie błędu
    loss = loss_fn(labels, pred_y)
    #zapisanie aktualnej wartości błędu
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(lr*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(lr*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(lr*dloss_dc)



Loss
```

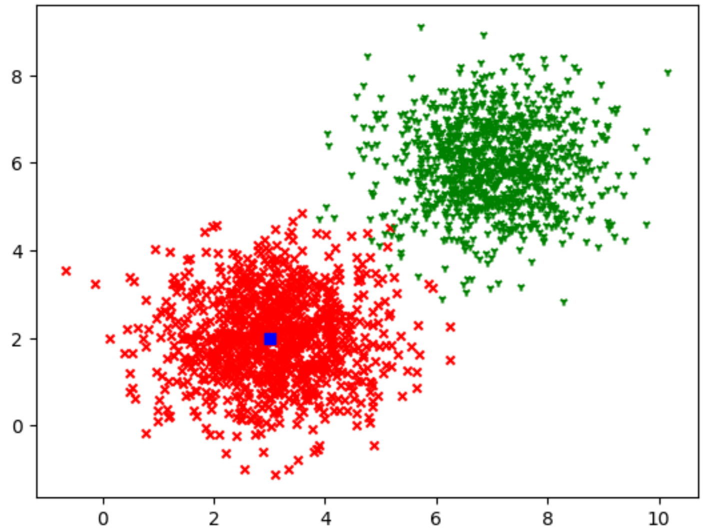
```
0.09580298,
...]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

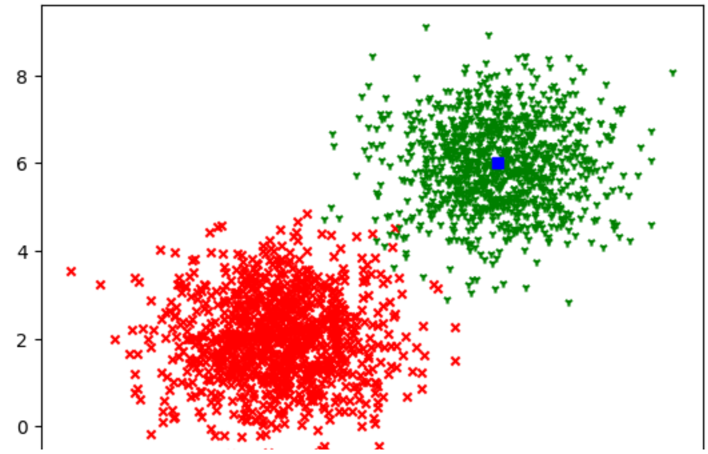



```
0.020424632
```

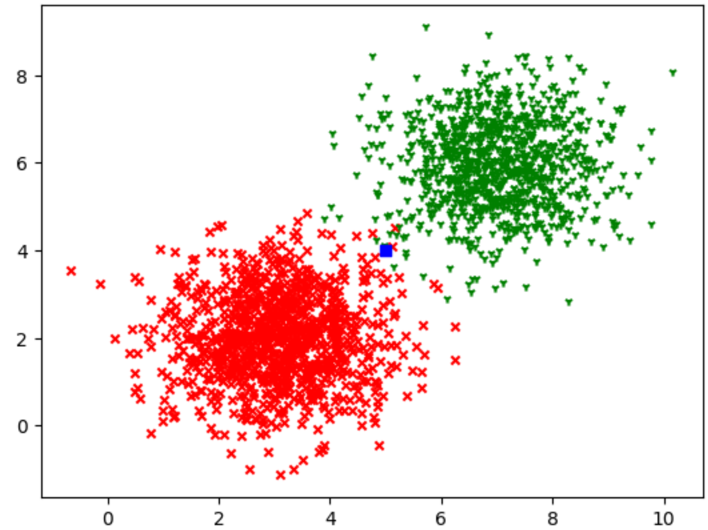
```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
0.020424632
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



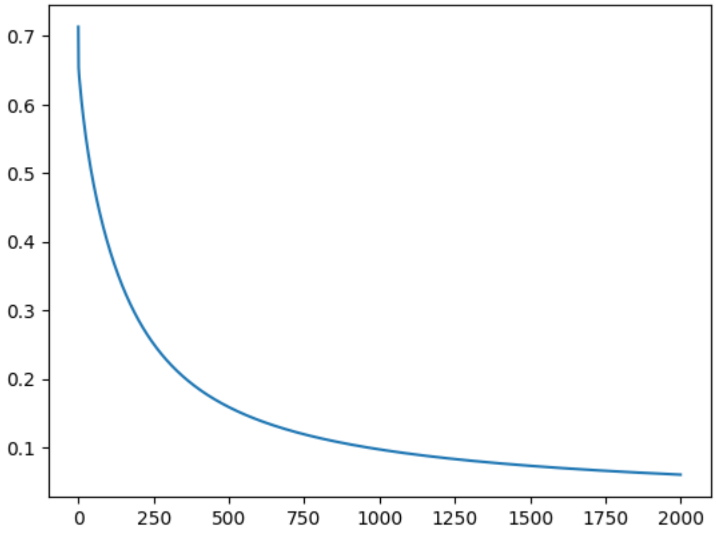
```
0.56779504
```

Ilość epok 2000

```
import random
a = tf.Variable(random.random())
b = tf.Variable(random.random())

c = tf.Variable(random.random())

Loss = []
epochs = 2000
lr = 0.1

for _ in range(epochs):

  with tf.GradientTape() as tape:
    #predykcja modelu

    pred_y = tf.sigmoid(a * xs + b * ys + c)
    #policzenie błędu
    loss = loss_fn(labels, pred_y)
    #zapisanie aktualnej wartości błędu
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(lr*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(lr*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(lr*dloss_dc)



Loss
```

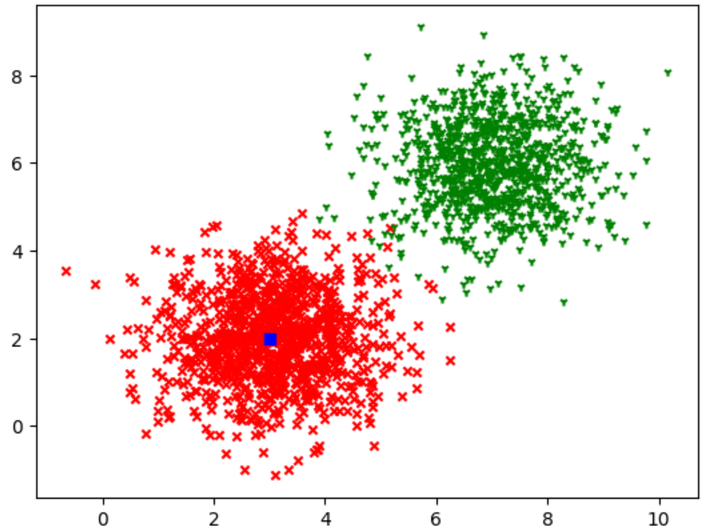
```
       0.09698636,
       ...]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

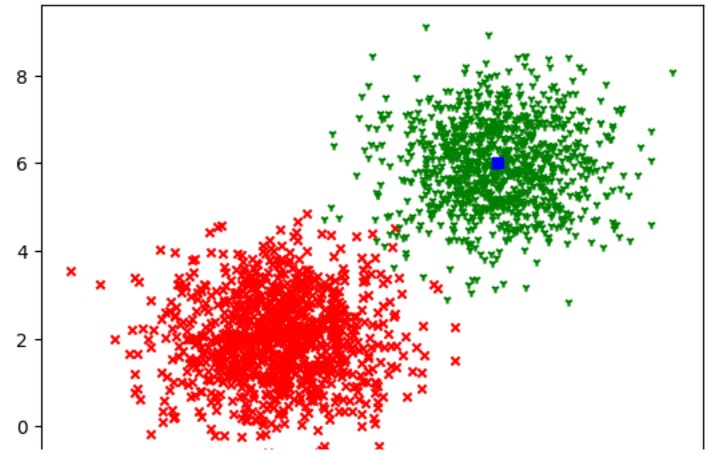


```
0.034258876
```

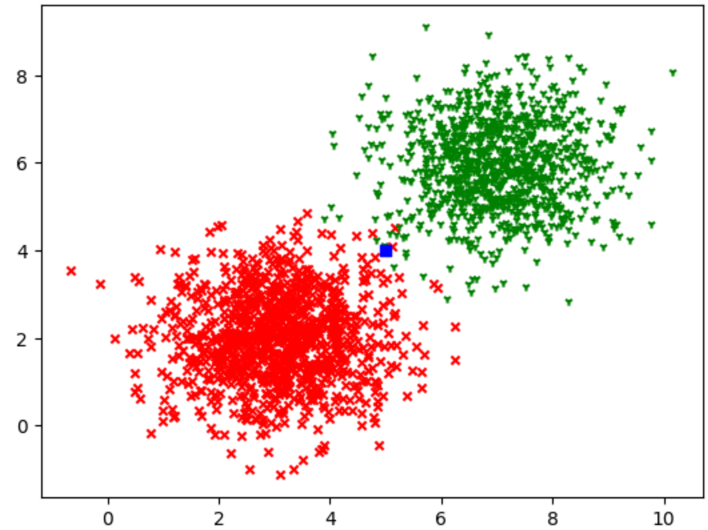
```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
0.034258876
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
0.5753893
```

Ilosc epok 100

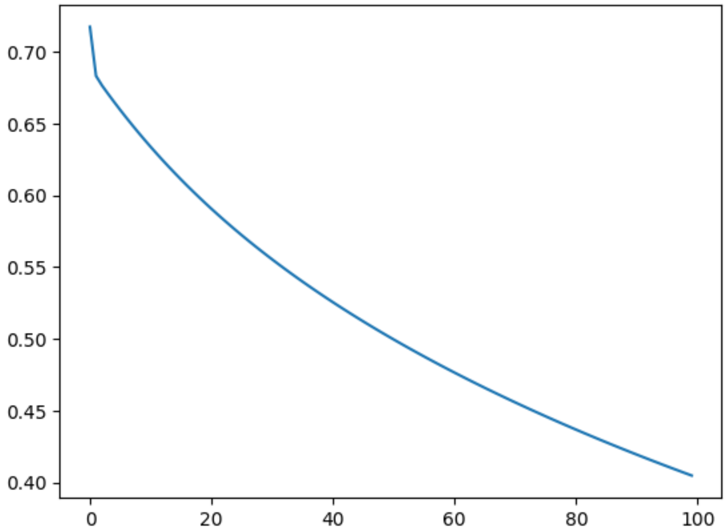
```
import random
a = tf.Variable(random.random())
b = tf.Variable(random.random())

c = tf.Variable(random.random())

Loss = []
epochs = 100
lr = 0.1

for _ in range(epochs):

  with tf.GradientTape() as tape:
    #predykcja modelu

    pred_y = tf.sigmoid(a * xs + b * ys + c)
    #policzenie błędu
    loss = loss_fn(labels, pred_y)
    #zapisanie aktualnej wartości błędu
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(lr*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(lr*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(lr*dloss_dc)
```

```
Loss
```

```
      0.51997524,
      0.51729715,
      0.5146541,
      0.512045,
      0.5094691,
      0.5069253,
      0.50441265,
      0.5019304,
      0.4994777,
      0.49705383,
      0.4946579,
      0.4922893,
      0.48994732,
      0.4876313,
```

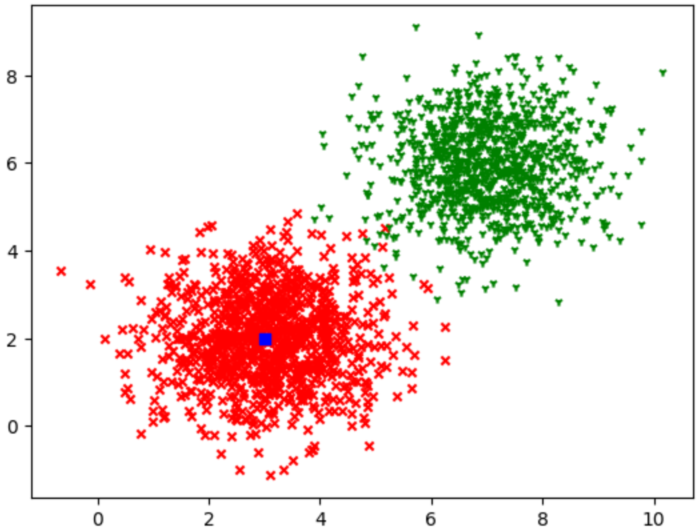
```
0.406291,
0.40472472]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

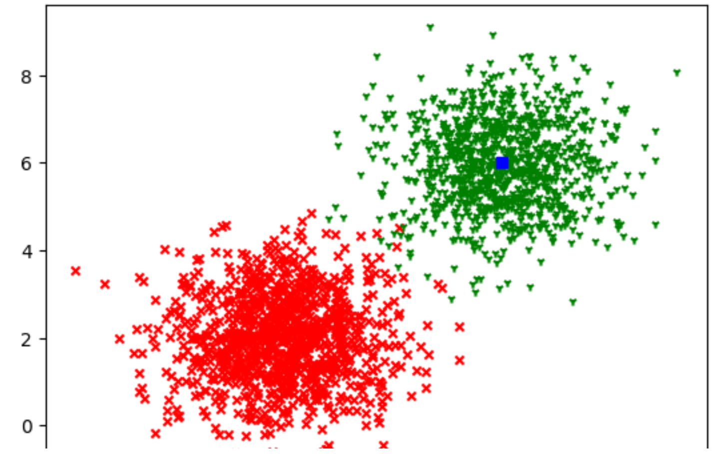



```
0.42606536
```

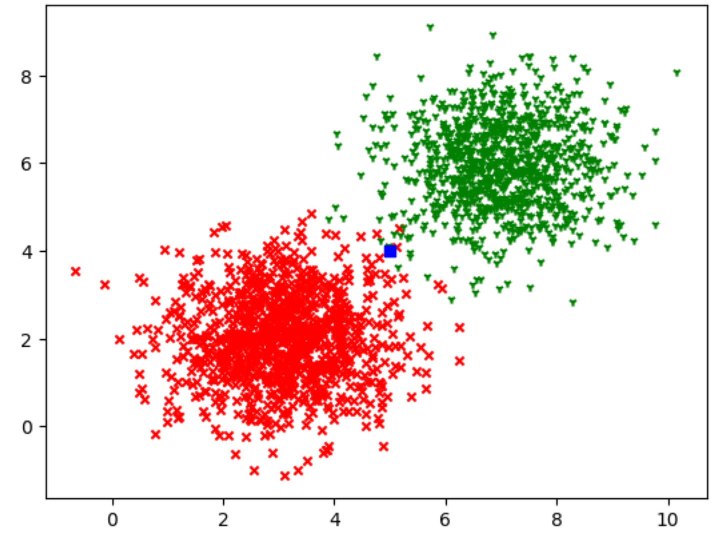
```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
0.42606536
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



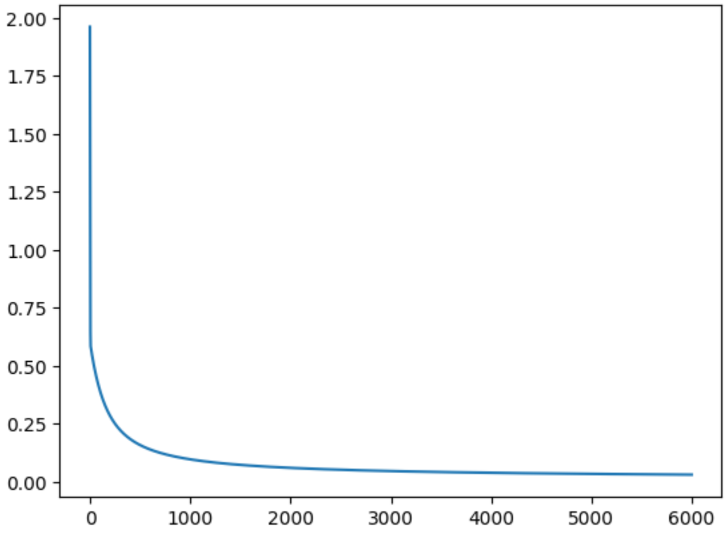
```
0.65546393
```

Ilosc epok - 6000

```
import random
a = tf.Variable(random.random())
b = tf.Variable(random.random())

c = tf.Variable(random.random())

Loss = []
epochs = 6000
lr = 0.1

for _ in range(epochs):

  with tf.GradientTape() as tape:
    #predykcja modelu

    pred_y = tf.sigmoid(a * xs + b * ys + c)
    #policzenie błędu
    loss = loss_fn(labels, pred_y)
    #zapisanie aktualnej wartości błędu
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(lr*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(lr*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(lr*dloss_dc)



Loss
```

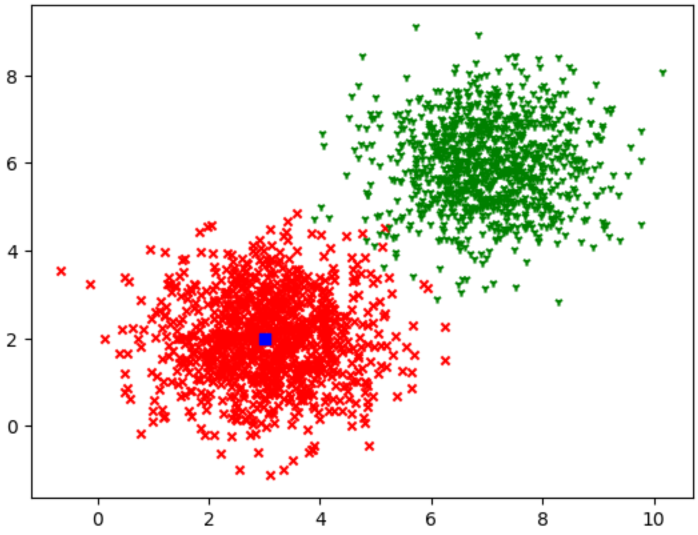
```
       0.09693862,
       ...]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

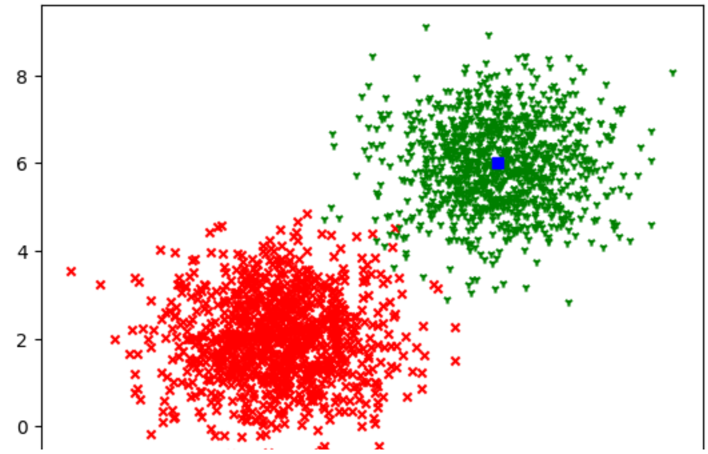



```
0.008089673
```

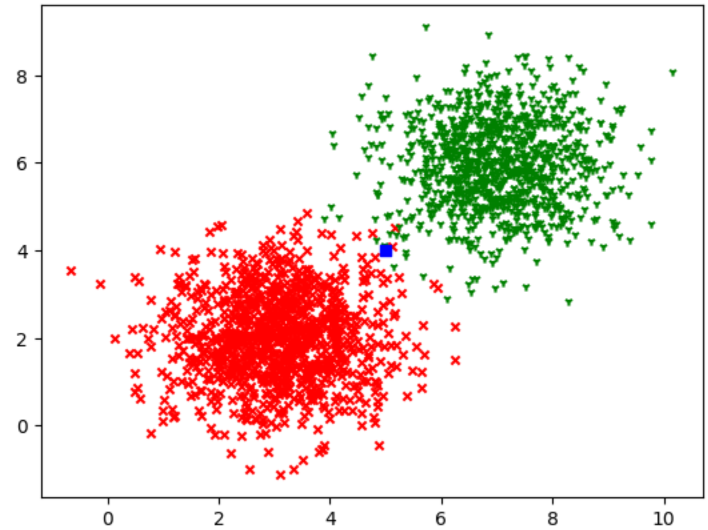
```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
0.008089673
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



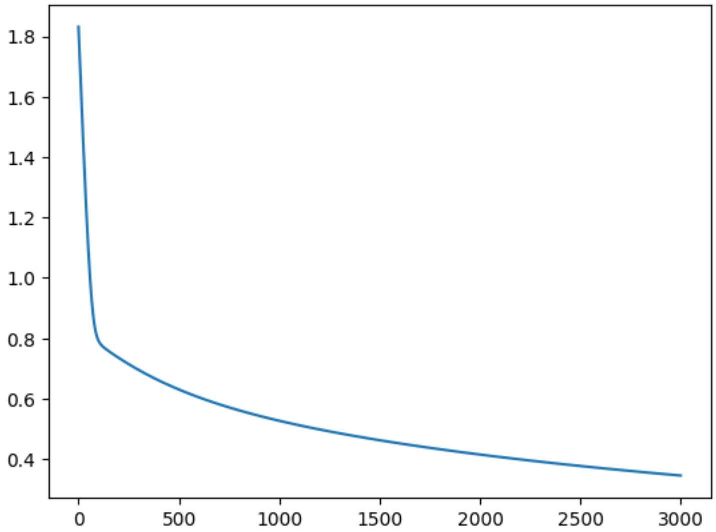
```
0.556591
```

Learning rate - 0.005

```
import random
a = tf.Variable(random.random())
b = tf.Variable(random.random())

c = tf.Variable(random.random())

Loss = []
epochs = 3000
lr = 0.005

for _ in range(epochs):

  with tf.GradientTape() as tape:
    #predykcja modelu

    pred_y = tf.sigmoid(a * xs + b * ys + c)
    #policzenie błędu
    loss = loss_fn(labels, pred_y)
    #zapisanie aktualnej wartości błędu
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(lr*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(lr*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(lr*dloss_dc)



Loss
```

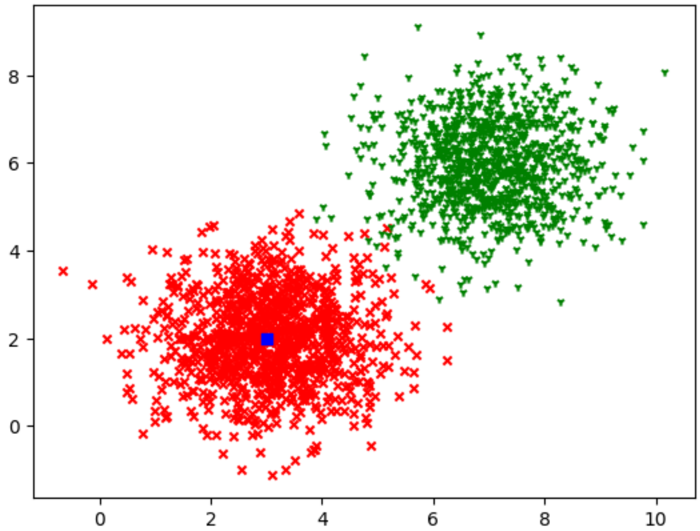
```
       0.5271256,
       ...]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

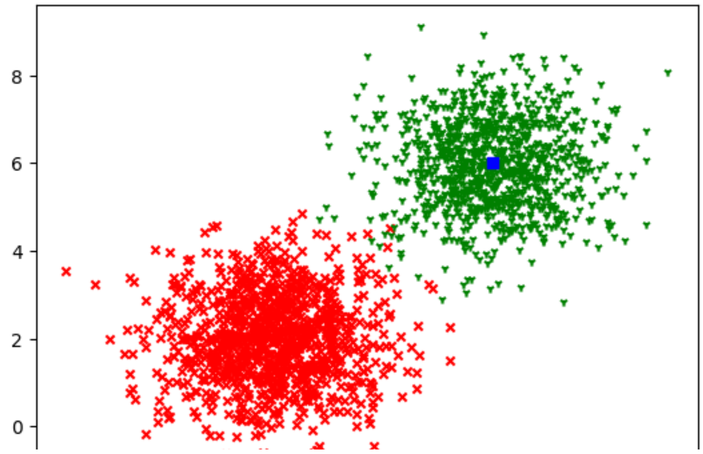



```
0.36911702
```

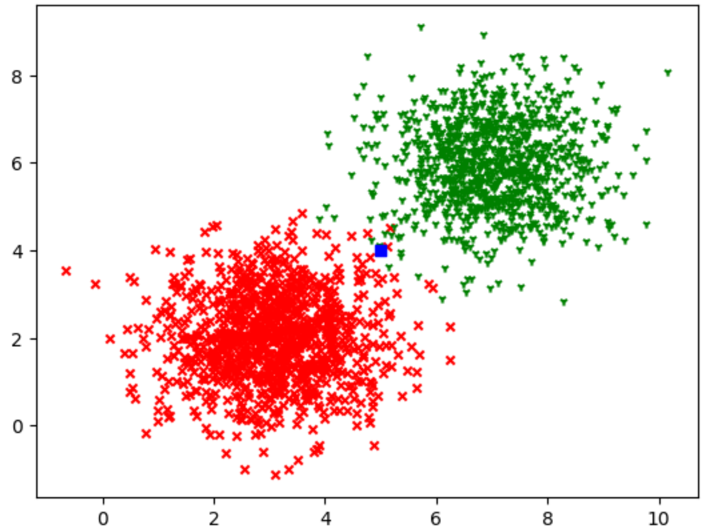
```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
0.36911702
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```




```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



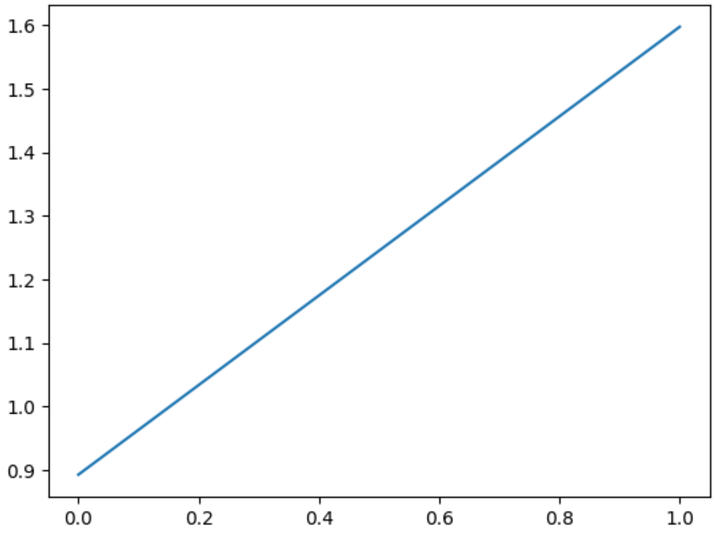

```
0.6452442
```

Learning rate - 0.5

```
import random
a = tf.Variable(random.random())
b = tf.Variable(random.random())

c = tf.Variable(random.random())

Loss = []
epochs = 3000
lr = 0.5

for _ in range(epochs):

  with tf.GradientTape() as tape:
    #predykcja modelu

    pred_y = tf.sigmoid(a * xs + b * ys + c)
    #policzenie błędu
    loss = loss_fn(labels, pred_y)
    #zapisanie aktualnej wartości błędu
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(lr*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(lr*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(lr*dloss_dc)



Loss
```

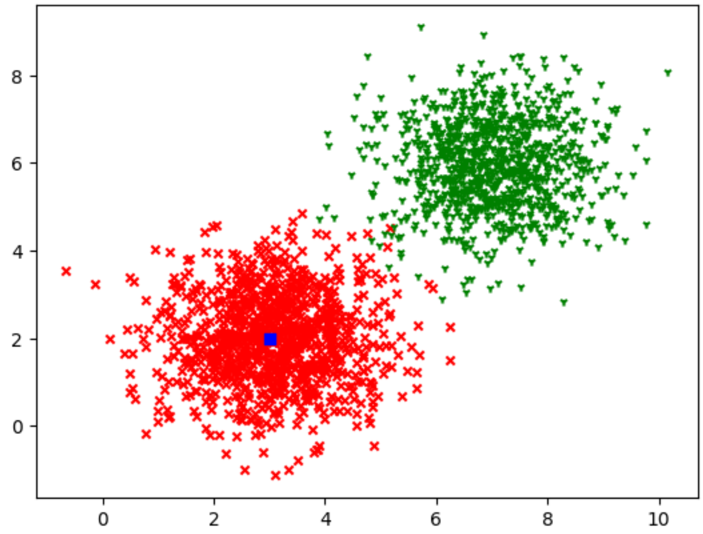
```
        nan,
        ...]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

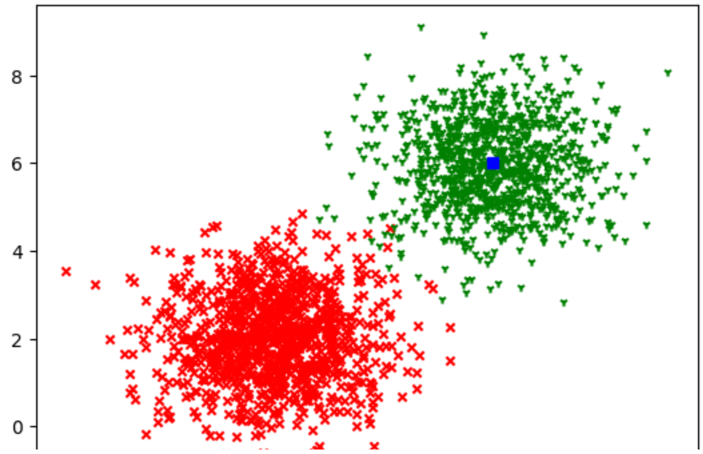


```
nan
```

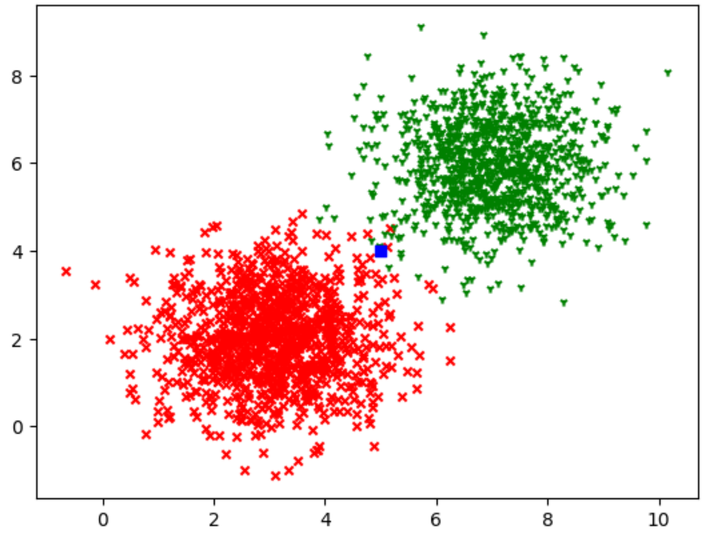
```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
nan
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```




```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```

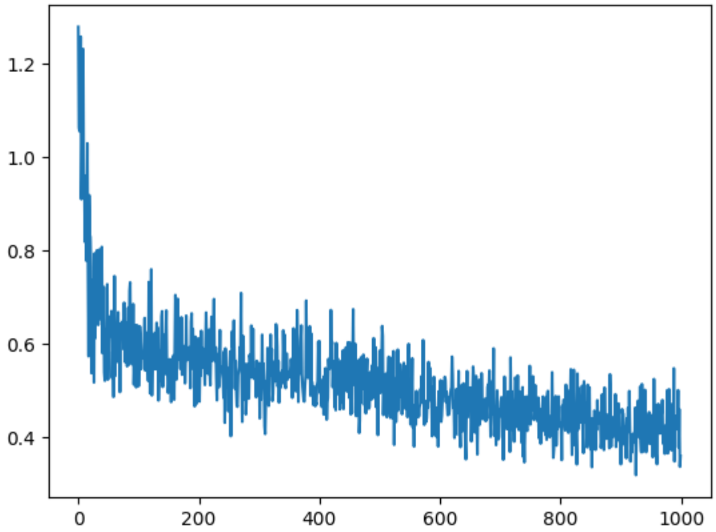


nan

Mini batch

```
def subset_dataset_2(x_dataset, y_dataset,label,subset_size):
    arr = np.arange(len(x_dataset))
    np.random.shuffle(arr)
    x_train = x_dataset[arr[0:subset_size]]
    y_train = y_dataset[arr[0:subset_size]]
    label_train = label[arr[0:subset_size]]
    return x_train,y_train,label_train
```

```
Loss = []
epochs = 1000
learning_rate = 0.01
batch_size = 50
a = tf.Variable(random.random())
b = tf.Variable(random.random())
c = tf.Variable(random.random())
for _ in range(epochs):
  xs_batch,ys_batch,labels_batch = subset_dataset_2(xs,ys,labels,batch_size)
  with tf.GradientTape() as tape:
    pred_l = tf.sigmoid(a * xs_batch + b * ys_batch + c)
    #print(label_batch.shape)
    loss = loss_fn(labels_batch, pred_l)
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(learning_rate*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(learning_rate*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(learning_rate*dloss_dc)


plt.plot(Loss)
plt.show()
```

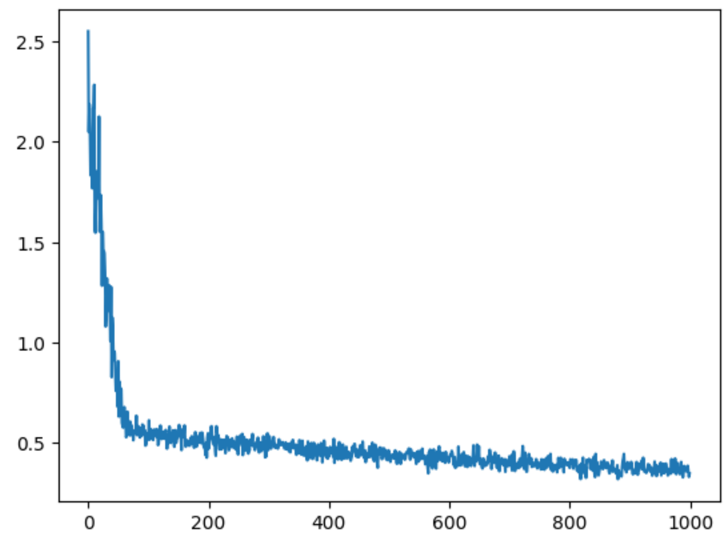


Mini batch - 100

```
Loss = []
epochs = 1000
learning_rate = 0.01
batch_size = 100
a = tf.Variable(random.random())
b = tf.Variable(random.random())
c = tf.Variable(random.random())
for _ in range(epochs):
  xs_batch,ys_batch,labels_batch = subset_dataset_2(xs,ys,labels,batch_size)
  with tf.GradientTape() as tape:
    pred_l = tf.sigmoid(a * xs_batch + b * ys_batch + c)
    #print(label_batch.shape)
    loss = loss_fn(labels_batch, pred_l)
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(learning_rate*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(learning_rate*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(learning_rate*dloss_dc)


Loss
```

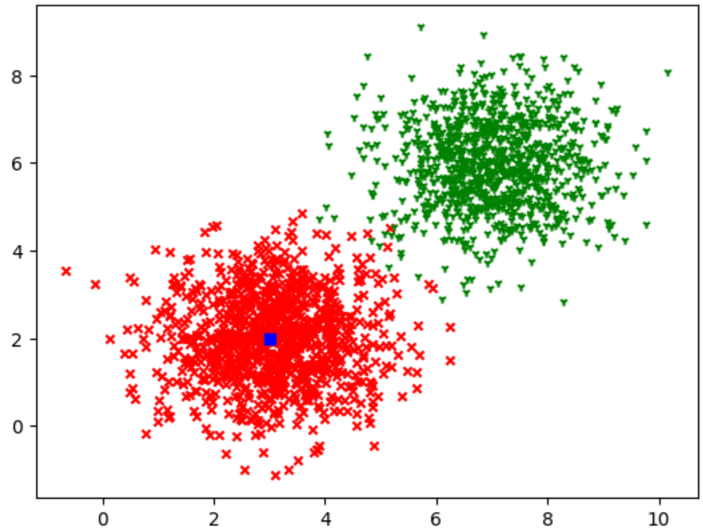
```
       0.3644971,
       0.37677056,
       0.40191123,
       0.33797646,
       0.3897355,
       0.35163426,
       0.3958945,
       0.37098607,
       0.34868005,
       0.35818943,
       0.39597285,
       0.39708325,
       0.37527192,
       0.37174198,
       0.3377356,
       0.37106168,
       0.36297703,
       0.36801812,
       0.34583023,
       0.36251992,
       0.41900155,
       0.3534843,
       0.38461724,
       0.36105064,
       0.35310227,
       0.37648743,
       0.4224778,
       0.3782796,
       0.3440729,
       0.37112737,
       0.39931995,
       0.38058147,
       0.35486615,
       0.3744294,
       0.34375858,
       0.34783754,
       0.40513542,
       0.36039096,
       0.3704394,
       0.3281107,
       0.35834983,
       0.38524735,
       0.377896,
       0.36143267,
       0.35872993,
       0.37753052,
       0.36032027,
       0.3860634,
       0.36042136,
       0.33057758,
       0.34787476]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

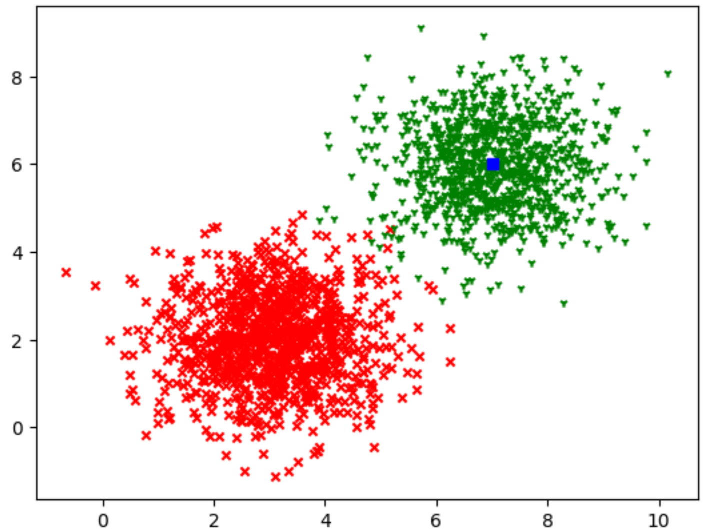


```
0.38763604
```

```
tf.sigmoid(a*x + b*y + c).numpy()
```

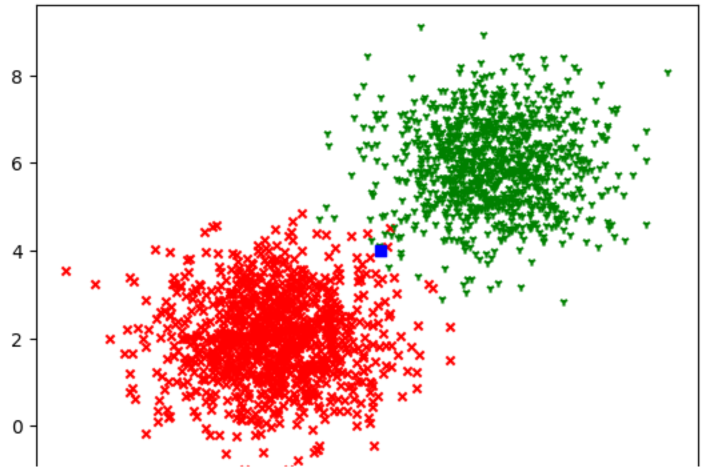
```
0.38763604
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
0.84294724
```

```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



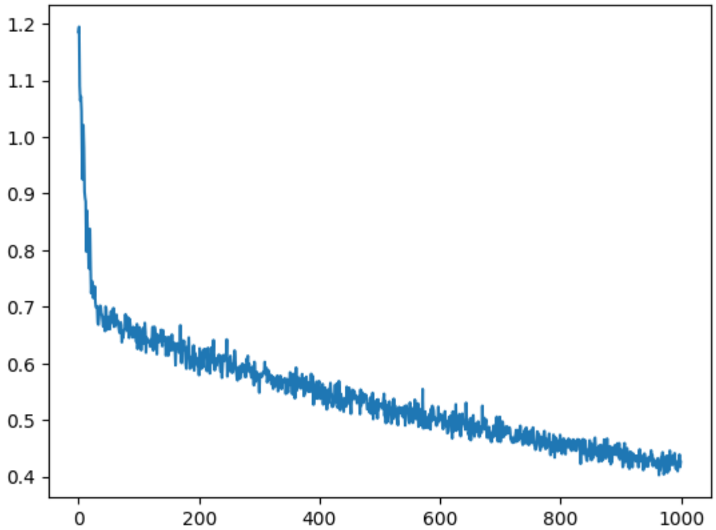
Mini batch - 500

```
Loss = []
epochs = 1000
learning_rate = 0.01
batch_size = 500
a = tf.Variable(random.random())
b = tf.Variable(random.random())
c = tf.Variable(random.random())
for _ in range(epochs):
  xs_batch,ys_batch,labels_batch = subset_dataset_2(xs,ys,labels,batch_size)
  with tf.GradientTape() as tape:
    pred_l = tf.sigmoid(a * xs_batch + b * ys_batch + c)
    #print(label_batch.shape)
    loss = loss_fn(labels_batch, pred_l)
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(learning_rate*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(learning_rate*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(learning_rate*dloss_dc)
```

```
Loss
```

```
0.40689993,
0.42887357,
0.43719596,
0.41100392,
0.44590193,
0.4268462,
0.42680717,
0.42477906,
0.4204055,
0.4367998,
0.42420018,
0.42430976,
0.4413236,
0.4198156,
0.41376477,
0.41646552,
0.41556278,
0.40958786,
0.42824462,
0.4206603,
0.43897077,
0.41698593,
0.42591131
```

```
plt.plot(Loss)
plt.show()
```



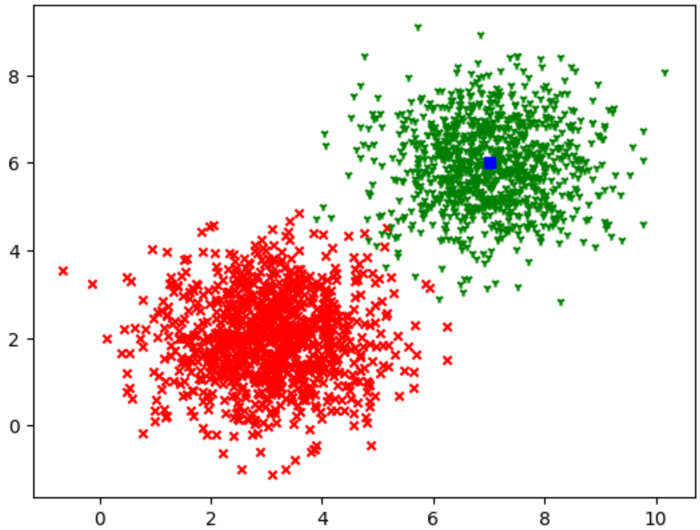
Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

```
tf.sigmoid(a*x + b*y + c).numpy()
```

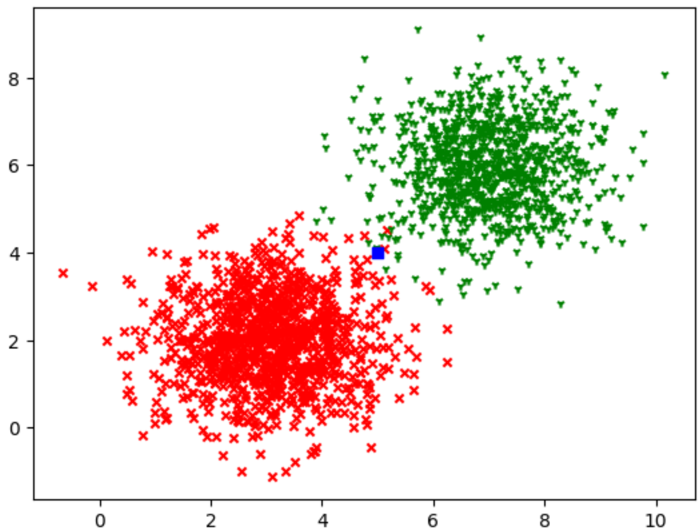
```
0.44046047
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
0.8249013
```

```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
0.65820616
```

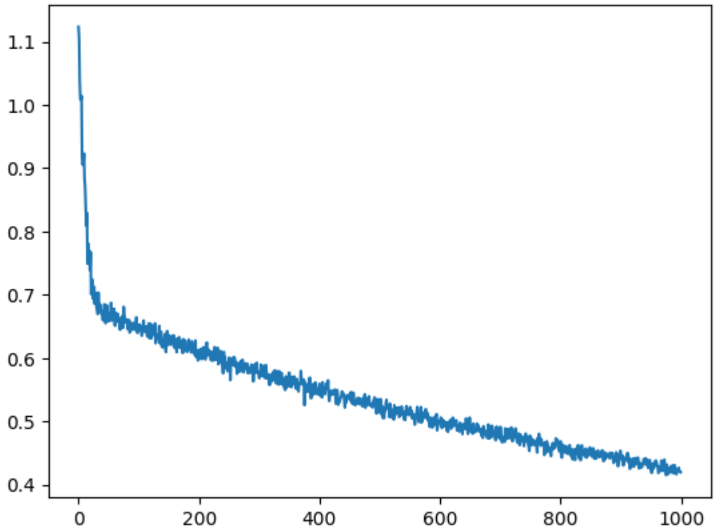
Mini batch - 1000

```
Loss = []
epochs = 1000
learning_rate = 0.01
batch_size = 1000
a = tf.Variable(random.random())
b = tf.Variable(random.random())
c = tf.Variable(random.random())
for _ in range(epochs):
  xs_batch,ys_batch,labels_batch = subset_dataset_2(xs,ys,labels,batch_size)
  with tf.GradientTape() as tape:
    pred_l = tf.sigmoid(a * xs_batch + b * ys_batch + c)
    #print(label_batch.shape)
    loss = loss_fn(labels_batch, pred_l)
    Loss.append(loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(learning_rate*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(learning_rate*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(learning_rate*dloss_dc)
```

```
Loss
```

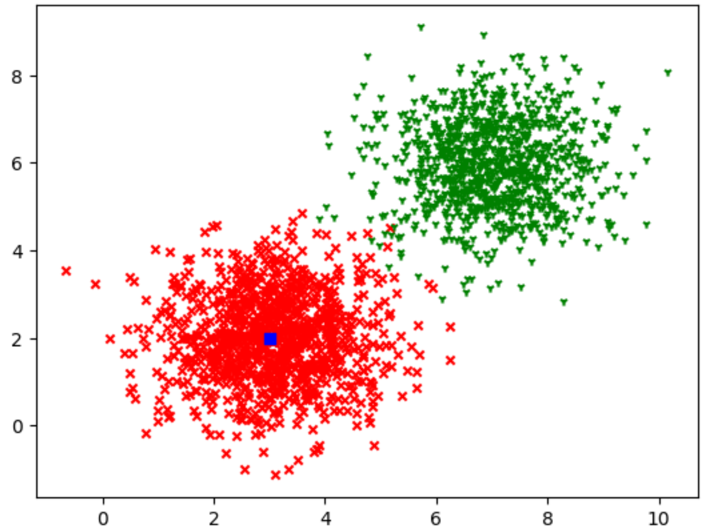
```
     0.43631133,
     0.43139648,
     0.43902272,
     0.42618427,
     0.4274808,
     0.43696985,
     0.43860143,
     0.4247756,
     0.42685205,
     0.418484,
     0.43226263,
     0.42838618,
     0.42495143,
     0.43426418,
     0.42888036,
     0.43734533,
     0.43795973,
     0.43478754,
     0.4306711,
     0.42837787,
     0.41828576,
     0.41924506,
     0.42841205,
     0.4285863,
     0.4230459,
     0.43790144,
     0.43286514,
     0.42135784,
     0.4322172,
     0.43681633,
     0.43948743,
     0.423264,
     0.43120423,
     0.41498664,
     0.4316234,
     0.4172738,
     0.42627335,
     0.41716278,
     0.42090997,
     0.42581177,
     0.4202376,
     0.42909986,
     0.41938856,
     0.42330298,
     0.43055415,
     0.42001525,
     0.41751075,
     0.4305068,
     0.42094296,
     0.42247534,
     0.4162971,
     0.42424417,
     0.42452207,
     0.42579523,
     0.4265987,
     0.42000666,
     0.4201903,
     0.41956922]
```

```
plt.plot(Loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

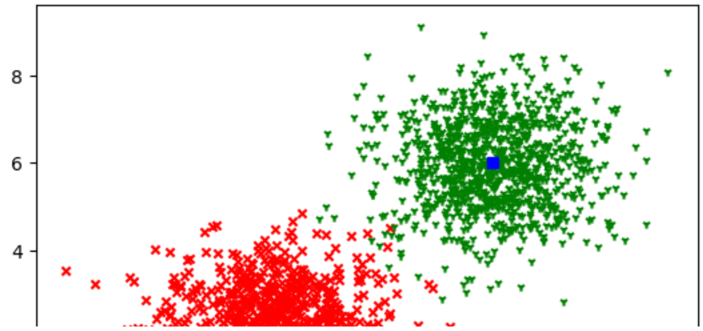



```
0.44105843
```

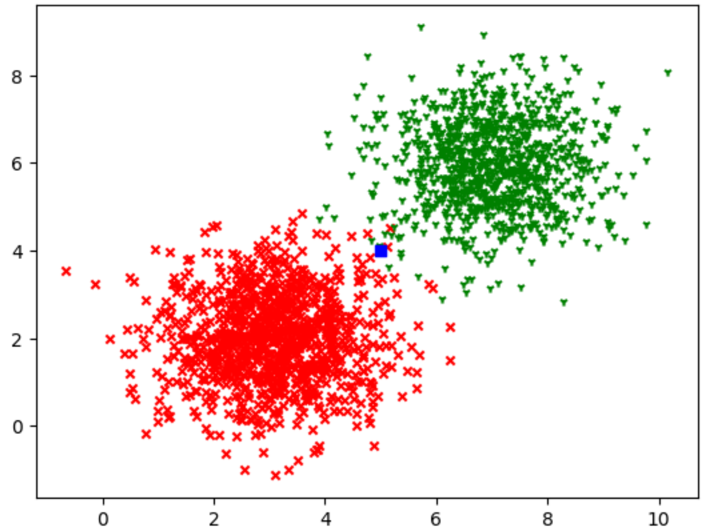
```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
0.44105843
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
0.6595454
```

Na uczenie modelu ma najwiekszy wpływ użycie batcha (bez batcha jest podawany cały zbiór uczący), dzięki temu wprowadza, że pewną losowość w procesie uczenia, pomoga to uniknąć utknięcia w minimach lokalnych. Model uczony z minibatchem osiąga lepsze rezultaty jeżeli chodzi o wyniki uczenia(szybszy spadek funkcji błędu oraz mniejszy błąd). Model lepiej i szybciej się uczy gdy mini-batch jest większy niż gdy jest on mniejszy.

Ponadto na proces uczenia modelu ma wpływ ilość epok. Za mała ilość epok skutkuje niedouczeniem modelu (model nie nauczył się wystarczająco dobrze dostosowywać się do danych treningowych). Ostatnim sprawdzonym przeze mnie parametrem jest współczynnik uczenia. Po przestawieniu na współczynnik Adam model uczy się lepiej Jego zbyt duża wartość rowadzi do skakania wokół minimum globalnego przy czym model go nie osiągnie. W przypadku zastosowania zbyt małej wartości współczynnika uczenia proces uczenia jest bardzo wolny na przełomie epok, a model "utyka" w minimach lokalnych.

Zad 3

```
def val_train_split(x_dataset, y_dataset,label,subset_size):
    arr = np.arange(len(x_dataset))
    l=len(x_dataset)
    split = int(len(x_dataset)*(1-subset_size))
    #print(split)
    np.random.shuffle(arr)
    x_train = x_dataset[arr[0:split]]
    y_train = y_dataset[arr[0:split]]
    label_train = label[arr[0:split]]
    x_val = x_dataset[arr[split:]]
    y_val = y_dataset[arr[split:]]
    label_val = label[arr[split:]]
    return x_train,y_train,label_train,x_val,y_val,label_val
```

```
x_train,y_train,label_train,x_val,y_val,label_val = val_train_split(xs,ys,labels,0.3)
```

```
x_train
```

```
array([6.91062517, 1.86620459, 7.16670689, ..., 6.66336346, 2.97810775,
       8.58583595])
```

```
y_train
```

```
array([5.2058503 , 1.63066471, 6.03677807, ..., 8.04114962, 1.99420774,
       6.11005225])
```

```
label_train
```

```
array([1., 0., 1., ..., 1., 0., 1.])
```

```
x_val
```

```
        4.23694609,  2.91191872,  6.0912127 ,  5.70518108,  8.86489112,
        5.94411632,  4.51912111,  5.37275078,  2.8167457 ,  8.44842017,
        4.81990471,  3.02350407,  6.91427337,  4.69000927,  2.101145  ,
        6.75936073,  3.08415557,  3.29051266,  1.84098755,  7.1696774 ,
        2.66655593,  7.35838444,  4.71377893,  3.63550263,  2.37783577,
        8.63173557,  2.78508198,  7.18862536,  1.10872565,  2.52166115,
        3.15585692,  7.4851557 ,  6.3676335 ,  4.82641085,  7.97824238,
        2.44365795,  7.82831203,  6.28704851,  5.14974123,  5.74784951,
        7.93088747,  1.85632838,  6.4829709 ,  6.78583236,  7.19470486,
        3.97958991,  9.08200539,  8.17546581,  8.5047958 ,  7.44453121,
        5.87826447,  5.59702182,  7.16837046,  2.12006002,  6.63933388,
        2.67798459,  8.12685975,  6.70738643,  2.86143356,  5.06511404,
        9.02763323,  3.63368754,  6.87508214,  7.60547582,  5.8100315 ,
        3.44737307,  2.81111461,  4.91056081,  1.94986937,  7.00444617,
        3.74224051,  5.06804891,  6.80014827,  3.13913316,  3.6203136 ,
        2.9540079 ,  4.01082403,  5.07502731,  7.45838013,  6.69834813,
        2.76201795,  8.44685572,  6.61285688,  2.46594098,  2.86484903,
        6.16700905,  2.90397817,  4.22771483,  1.70952092,  7.82374281,
        6.80197321,  6.8117379 ,  7.83189026,  6.42669079,  3.87656615,
        3.34273807,  2.79649021,  6.01848158,  3.19896792,  7.44614514,
        3.45647553,  2.3289811 ,  4.24423223,  3.9094234 ,  7.74150088,
        8.61332315,  5.5964156 ,  1.49499143,  6.98556966,  6.92131532,
        3.5712222 ,  2.73321808,  7.14313807,  3.52723391,  7.51479919,
        6.28598755,  7.01427886,  6.65558955,  7.76756807,  1.59913118,
        4.92994678,  8.35095865,  7.20508928,  3.37596598,  5.82691756,
        5.08021014,  6.55038007,  7.83112397,  3.22928966,  3.66980601,
        4.90141626,  6.72304854,  3.02026616,  3.93241336,  6.33215033,
        2.88320561,  3.27251291,  7.02718079,  1.49940707,  3.35819982,
        1.85314964,  7.69470293,  7.55502122,  2.98064708,  7.20510279,
        1.10518237,  4.09480348,  2.72496657,  8.57422408,  4.15638057,
        7.36366368,  4.80117851,  5.64832464,  5.81387501,  4.8453065 ,
        4.03726593,  2.49845719,  3.32018095,  5.62988135,  5.24235881,
        8.03289982,  1.6834958 ,  7.79806985,  3.79015445,  6.25049221,
        3.62594723,  4.42028074,  2.5224419 ,  6.31636402,  6.21600662,
        6.65977894,  3.75226202,  3.20682518,  3.27439361,  5.96233354,
        0.78392538,  7.3375129 ,  3.57786424,  3.13436021,  7.03321665,
        5.658583  ,  4.04731204,  2.92029948,  7.00567891,  3.16264786,
        7.93808692,  5.63447072,  6.24337001,  2.50803228,  1.7623002 ,
        7.2043096 ,  6.35729188,  2.20558531,  3.00741247,  6.79068139,
        4.39269078,  2.21406522,  3.68392228,  3.15671941,  6.77538828,
        7.39704329,  6.91858995,  7.14708798,  1.74233449,  2.49664437,
        2.27444626, -0.14026429,  7.20499643,  6.47223703,  2.91237916,
        4.35705282,  3.09189768,  6.48034276,  5.83099128,  3.18776237,
        4.84198757,  2.04315866,  8.3608385 ,  2.67063246,  2.09726167,
        7.15818386,  2.04700681,  5.67658673,  7.35533256,  6.33601599,
        7.94557774,  4.05935818,  3.50225365,  8.58454067,  5.35655785,
        3.45079445,  3.70303089,  3.88225816,  2.53114841,  3.80024565,
        4.07030286,  7.82737085,  1.01532056,  3.45480824,  4.3632894 ,
        7.69797158,  4.60807321,  6.54354876,  2.22801172,  2.36381613,
        7.53953759,  6.34351408,  6.48938576,  3.78289686,  1.83558853,
        3.20395532,  6.79032428,  8.49446895,  8.51148023,  4.20950751,
        7.3371662 ,  7.33051474,  5.64669623,  3.64537087,  6.66108337,
        3.00428296,  2.84711054,  3.61282705,  7.72362696,  3.36517355,
        2.41048018,  6.99421566,  1.21861423,  6.12147182,  7.7357632 ,
        3.46873568,  8.13685489,  1.97649507,  6.24991713,  7.02707294,
        3.66144264,  7.81397427,  3.3238234 ,  4.84576016,  6.43853202,
        6.82956582,  9.76113332,  6.3393887 ,  7.69642098,  6.52329306,
        4.03716165,  2.80238311,  2.15842014,  2.84545068,  3.20545672])
```

```
y_val
```

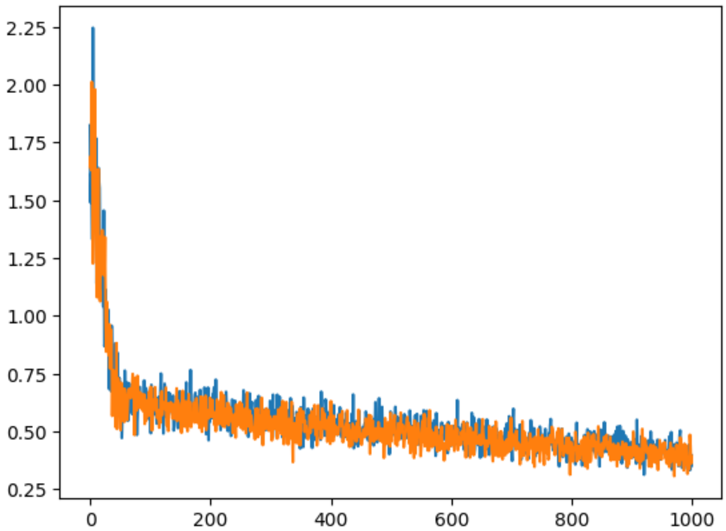
```
        4.09583382e+00,  2.66168662e+00,  1.55670157e+00,  4.79024344e+00,
        5.14177868e-01,  6.35847706e-01,  1.67786402e+00,  7.25749170e+00,
        5.62376841e+00,  7.32547782e+00,  7.58707903e+00,  7.09544461e+00,
       -5.21137703e-01,  1.65410461e+00,  3.99467882e+00,  5.18873694e+00,
        1.28603565e+00,  5.17399510e+00,  2.10768938e+00,  1.80904199e+00,
        3.51095658e-01,  1.38306085e+00,  4.63415280e+00,  5.24120884e+00,
        6.49862817e+00,  6.78239858e-01,  5.83215975e+00,  5.75405946e+00,
        1.14032911e+00,  2.27319357e+00,  7.03774862e+00,  2.23984355e+00,
        3.18865412e+00,  5.43353592e+00,  6.30411269e+00,  5.79333176e+00,
        5.90989796e+00,  2.62467807e+00,  7.02653561e+00,  7.89089572e+00,
        7.57967702e+00,  2.79258118e+00,  6.22351909e+00,  1.82936025e+00,
        3.37888185e+00,  6.13260128e+00,  1.57118241e+00,  2.38137832e+00,
        1.48486280e+00,  4.90651764e+00,  1.65950070e+00,  1.51783372e+00,
        7.43386080e+00,  2.23129562e+00,  2.70311074e+00,  4.54939613e+00,
        1.98005170e+00,  3.22862015e+00,  1.62907263e+00,  7.17246962e+00,
        5.95908800e+00,  1.05556550e+00,  6.29638886e+00,  8.50338789e-01,
        2.33564656e+00,  1.72746606e+00,  6.06434089e+00,  1.75270649e+00,
        4.82567608e+00,  8.71320162e-02,  1.23955250e+00,  7.07502001e+00,
        4.34448057e-01,  2.92364236e+00,  2.05120282e+00,  2.41700127e+00,
        4.60219501e+00,  4.55315007e+00,  5.93196809e+00,  1.81220693e+00,
        5.06423146e+00,  2.07346970e+00,  6.50964240e+00,  1.86300278e+00,
        2.50318270e+00,  2.78909918e+00,  7.61113595e+00,  6.45338418e+00,
        5.95519509e+00,  2.28649949e+00,  2.05781992e+00,  2.66506987e+00,
        6.94127531e+00,  2.87076846e+00,  4.48837486e+00,  1.06742733e+00,
        4.12038891e+00,  7.22293978e+00,  5.63657543e+00,  2.63860025e+00,
        2.25734972e+00,  7.51472156e+00,  2.14991896e+00,  5.37399947e+00,
        5.23085486e+00,  5.31539456e+00,  1.37516722e-01,  2.26303062e+00,
        5.30451521e+00,  6.89387085e+00, -6.23922190e-01,  2.03792432e+00,
        8.29309505e+00,  3.61149968e-01,  1.30497531e+00,  1.03407953e+00,
        2.93409456e-01,  5.90386230e+00,  7.54874256e+00,  5.75196236e+00,
        6.24965685e+00,  2.70146968e+00,  1.57697034e+00,  3.01408259e+00,
        3.23248300e+00,  7.28369052e+00,  4.69427033e+00,  1.18365785e+00,
        1.86687794e+00,  3.79955820e+00,  7.51107687e+00,  6.38283801e+00,
        1.06622008e+00,  5.28949605e+00,  1.99625106e+00,  6.20507002e+00,
        2.69560360e+00,  2.88760367e+00,  7.03096805e+00,  1.94884661e+00,
        6.44190181e+00,  4.45431892e+00,  6.60657263e+00,  6.92444337e+00,
        2.22537405e+00,  2.25564722e+00,  6.78415732e+00,  3.97424022e+00,
        2.93374217e+00,  3.19360015e+00,  1.08233401e+00,  2.93422995e+00,
        7.52529729e-01,  3.89079325e+00,  5.18649550e+00,  5.87310870e-01,
        1.32198227e+00,  1.77839653e+00,  6.37079104e+00,  2.39623031e+00,
        5.40857970e+00,  1.40481533e+00,  2.01982939e+00,  6.82496979e+00,
        5.06765712e+00,  7.47075397e+00,  3.41261940e+00,  2.21730097e+00,
        1.14327894e+00,  6.98943376e+00,  6.90996821e+00,  7.19881115e+00,
        1.13514595e+00,  5.33989743e+00,  5.23668805e+00,  6.74665698e+00,
        9.49592949e-01,  4.29129580e+00,  1.40545277e+00,  1.85960265e+00,
        2.10899633e+00,  5.04394842e+00,  3.38686292e+00,  1.35942285e+00,
        6.86905636e+00,  2.73380973e+00,  4.25571356e+00,  6.96747522e+00,
        1.88126941e+00,  5.91671958e+00,  5.50342675e-01,  2.28110512e+00,
        7.59724760e+00,  2.80363205e+00,  5.50737696e+00,  1.31309762e+00,
        2.61362848e+00,  5.55329887e+00,  7.38661289e+00,  6.08458955e+00,
        5.42463473e+00,  7.96990411e+00,  3.05453946e+00,  6.67023727e+00,
        1.05707159e+00,  1.18291146e+00,  3.94751234e+00,  2.17371789e+00])
```

```
label_val
```

```
array([0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0.,
       0., 1., 1., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 1., 1.,
       1., 1., 0., 0., 1., 1., 1., 0., 0., 0., 1., 1., 1., 0., 0., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 0., 0.,
       1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0.,
       0., 0., 1., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 1.,
       0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 1., 1., 1.,
       0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 0.,
       0., 0., 1., 0., 0., 0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 0., 1.,
       1., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1.,
       0., 0., 0., 1., 1., 1., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1.,
       0., 0., 0., 1., 0., 0., 1., 1., 1., 0., 0., 0., 0., 0., 1., 1., 0.,
       0., 0., 1., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 1.,
       1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0.,
       0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 0., 1., 0., 1.,
       0., 0., 1., 0., 1., 0., 1., 0., 1., 0., 0., 0., 1., 1., 1., 0., 1.,
       1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 0., 1.,
       1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1.,
       0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0.,
       0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1., 1.,
       1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0., 1., 1., 0.,
       1., 1., 0., 0., 1., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0., 1.,
       0., 1., 0., 0., 0., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1., 0., 0., 0., 1.,
       0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1.,
       0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 1., 0., 1., 0., 1., 0.,
       0., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0.,
       1., 0., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 1.,
       1., 1., 1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 1., 0., 1., 0.,
       1., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 1., 1., 0., 0., 0., 0.,
       0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 1., 0., 0., 0.,
       1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0.,
```

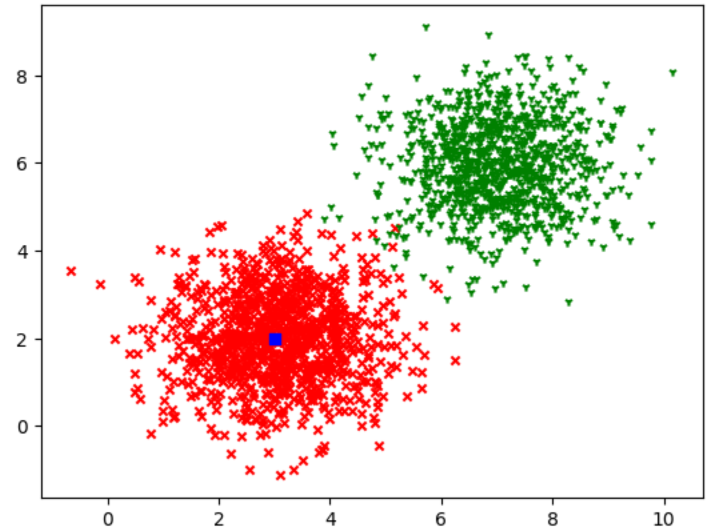
```
           1., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 1., 1., 1., 1., 1., 1.,
           1., 0., 0., 0., 0.])
```

```
def subset_dataset_2(x_dataset, y_dataset,label,subset_size):
    arr = np.arange(len(x_dataset))
    np.random.shuffle(arr)
    x_train = x_dataset[arr[0:subset_size]]
    y_train = y_dataset[arr[0:subset_size]]
    label_train = label[arr[0:subset_size]]
    return x_train,y_train,label_train


Loss = []
Val_loss =[]
epochs = 1000
learning_rate = 0.01
batch_size = 50
val_split = 0.3
a = tf.Variable(random.random())
b = tf.Variable(random.random())
c = tf.Variable(random.random())
x_tr,y_tr,label_tr,x_val,y_val,label_val = val_train_split(xs,ys,labels,val_split)
for _ in range(epochs):
  xs_batch,ys_batch,labels_batch = subset_dataset_2(x_tr,y_tr,label_tr,batch_size)
  xsv_batch,ysv_batch,labelsv_batch = subset_dataset_2(x_val,y_val,label_val,batch_size)
  with tf.GradientTape() as tape:
    pred_l = tf.sigmoid(a * xs_batch + b * ys_batch + c)
    pred_lv = tf.sigmoid(a * xsv_batch + b * ysv_batch + c)
    #print(label_batch.shape)
    loss = loss_fn(labels_batch, pred_l)
    val_loss = loss_fn(labelsv_batch,pred_lv)
    Loss.append(loss.numpy())
    Val_loss.append(val_loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(learning_rate*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(learning_rate*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(learning_rate*dloss_dc)



plt.plot(Loss)
plt.plot(Val_loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

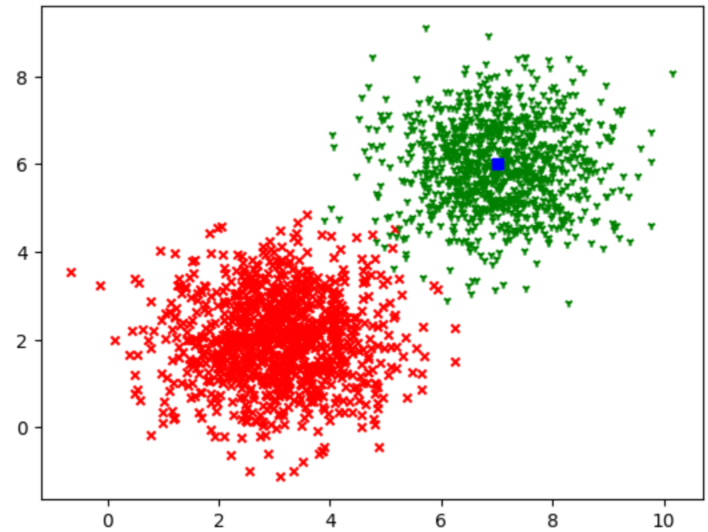


```
0.44703996
```

```
tf.sigmoid(a*x + b*y + c).numpy()
```

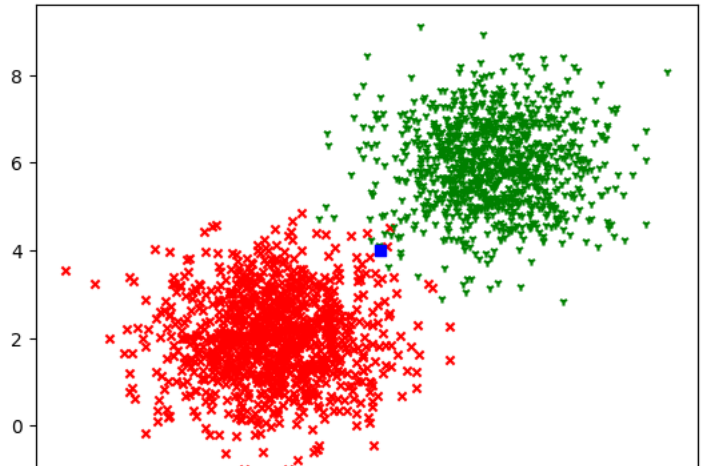
```
0.44703996
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



```
0.8512708
```

```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



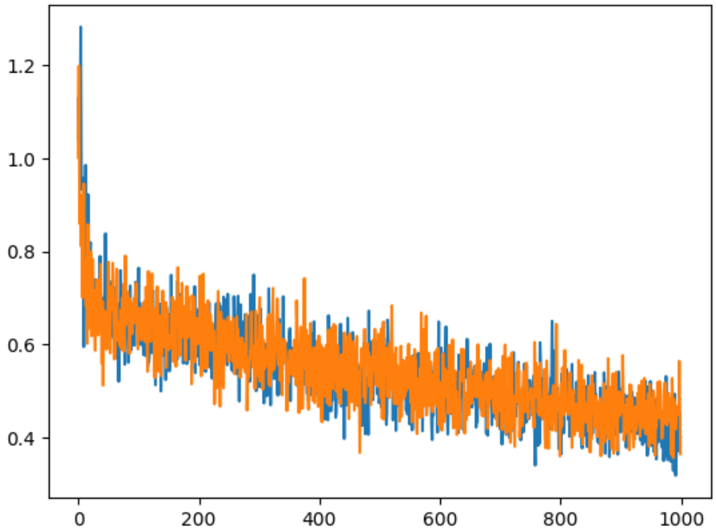
Val_split - 0.4

```
Loss = []
Val_loss =[]
epochs = 1000
learning_rate = 0.01
batch_size = 50
val_split = 0.4
a = tf.Variable(random.random())
b = tf.Variable(random.random())
c = tf.Variable(random.random())
x_tr,y_tr,label_tr,x_val,y_val,label_val = val_train_split(xs,ys,labels,val_split)
for _ in range(epochs):
  xs_batch,ys_batch,labels_batch = subset_dataset_2(x_tr,y_tr,label_tr,batch_size)
  xsv_batch,ysv_batch,labelsv_batch = subset_dataset_2(x_val,y_val,label_val,batch_size)
  with tf.GradientTape() as tape:
    pred_l = tf.sigmoid(a * xs_batch + b * ys_batch + c)
    pred_lv = tf.sigmoid(a * xsv_batch + b * ysv_batch + c)
    #print(label_batch.shape)
    loss = loss_fn(labels_batch, pred_l)
    val_loss = loss_fn(labelsv_batch,pred_lv)
    Loss.append(loss.numpy())
    Val_loss.append(val_loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(learning_rate*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(learning_rate*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(learning_rate*dloss_dc)
```

```
Loss
```

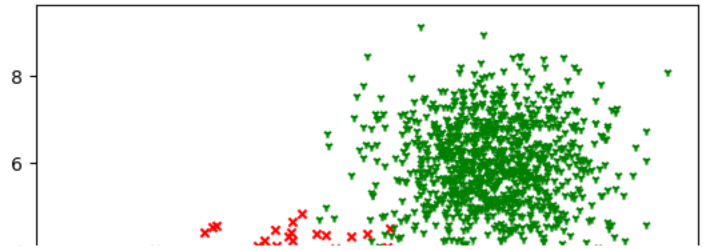
```
       0.38837147,
       0.37817046,
       0.45636854,
       0.35064736,
       0.38606307,
       0.3596329,
       0.42598155,
       0.5318419,
       0.42551738,
       0.47840893,
       0.35215783,
       0.43179715,
       0.40277427,
       0.41627347,
       0.3937735,
       0.34894252,
       0.39706868,
       0.48803025,
       0.3864722,
       0.32954022,
       0.40705982,
       0.49416503,
       0.40349248,
       0.31906512,
       0.35690677,
       0.40477276,
       0.46485162,
       0.39797634,
       0.37145865,
       0.45478436,
       0.39694333,
       0.40097886]
```

```
plt.plot(Loss)
plt.plot(Val_loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

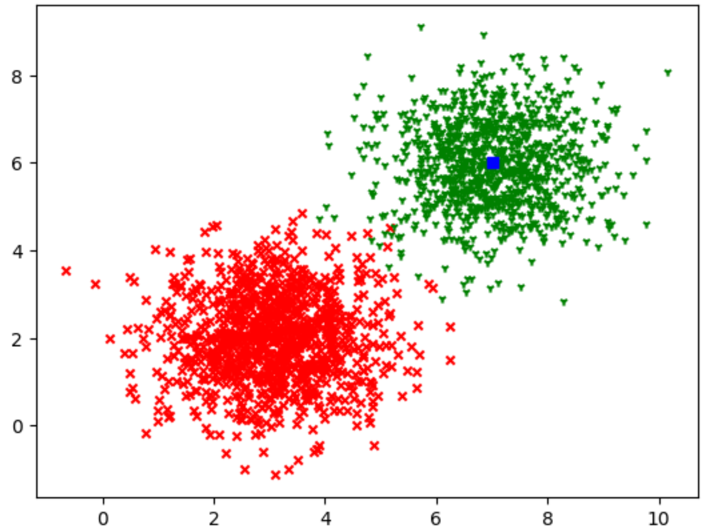
```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```




```
tf.sigmoid(a*x + b*y + c).numpy()
```

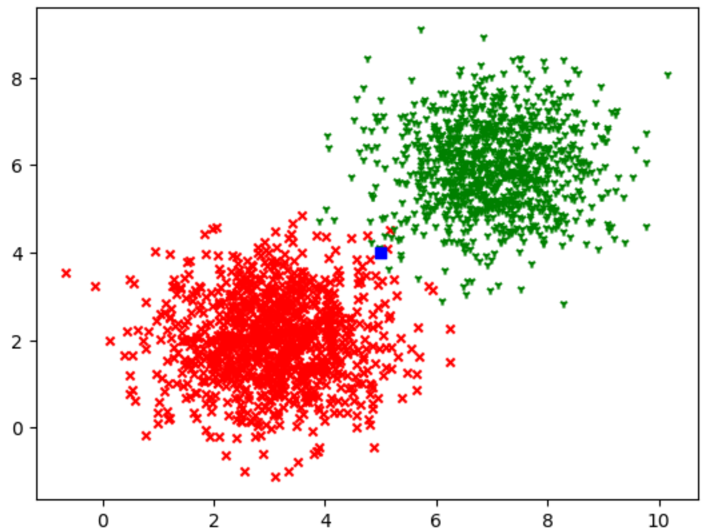
0.44605067

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



0.83358884

```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```



0.6675931

Aby edytować zawartość komórki, kliknij ją dwukrotnie (lub naciśnij klawisz Enter)

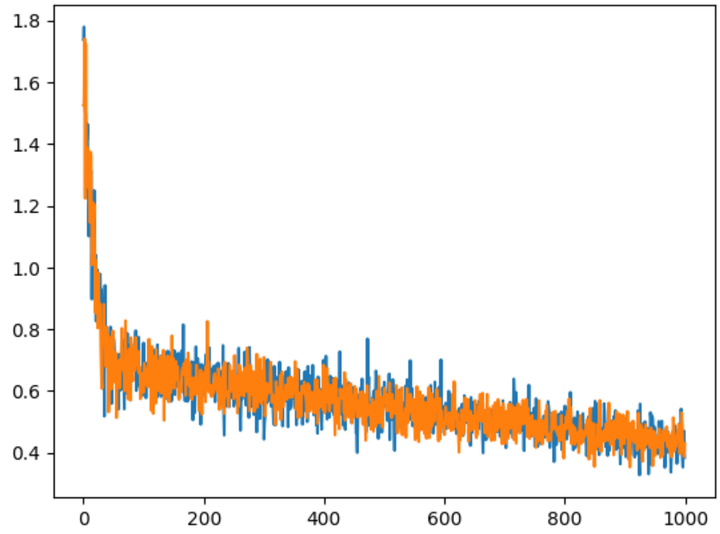
Val_split - 0.1

```
Loss = []
Val_loss =[]
epochs = 1000
learning_rate = 0.01
batch_size = 50
val_split = 0.1
a = tf.Variable(random.random())
b = tf.Variable(random.random())
c = tf.Variable(random.random())
x_tr,y_tr,label_tr,x_val,y_val,label_val = val_train_split(xs,ys,labels,val_split)
for _ in range(epochs):
  xs_batch,ys_batch,labels_batch = subset_dataset_2(x_tr,y_tr,label_tr,batch_size)
  xsv_batch,ysv_batch,labelsv_batch = subset_dataset_2(x_val,y_val,label_val,batch_size)
  with tf.GradientTape() as tape:
    pred_l = tf.sigmoid(a * xs_batch + b * ys_batch + c)
    pred_lv = tf.sigmoid(a * xsv_batch + b * ysv_batch + c)
    #print(label_batch.shape)
    loss = loss_fn(labels_batch, pred_l)
    val_loss = loss_fn(labelsv_batch,pred_lv)
    Loss.append(loss.numpy())
    Val_loss.append(val_loss.numpy())

  dloss_da, dloss_db, dloss_dc = tape.gradient(loss,(a, b,c))

  a.assign_sub(learning_rate*dloss_da)  #a = a - alpha*dloss_da
  b.assign_sub(learning_rate*dloss_db)  #b = b - alpha*dloss_db
  c.assign_sub(learning_rate*dloss_dc)
```

```
Loss
```

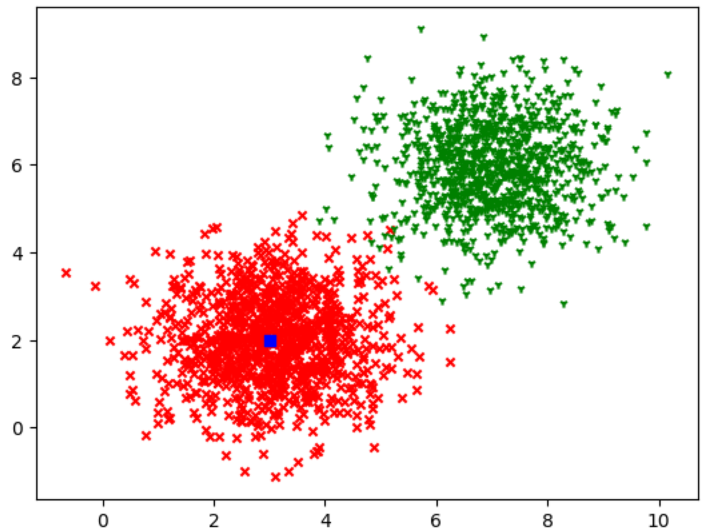
```
0.40273422,
0.49227914,
0.4793527,
0.4120394,
0.5415535,
0.48539612,
0.3821492,
0.3525137,
0.46637332,
0.47075132,
0.38399902]
```

```
plt.plot(Loss)
plt.plot(Val_loss)
plt.show()
```



Sprawdzamy dla pewnego punktu:

```
x=3.0
y=2.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
#print(a,b,c)
tf.sigmoid(a*x + b*y + c).numpy()
```

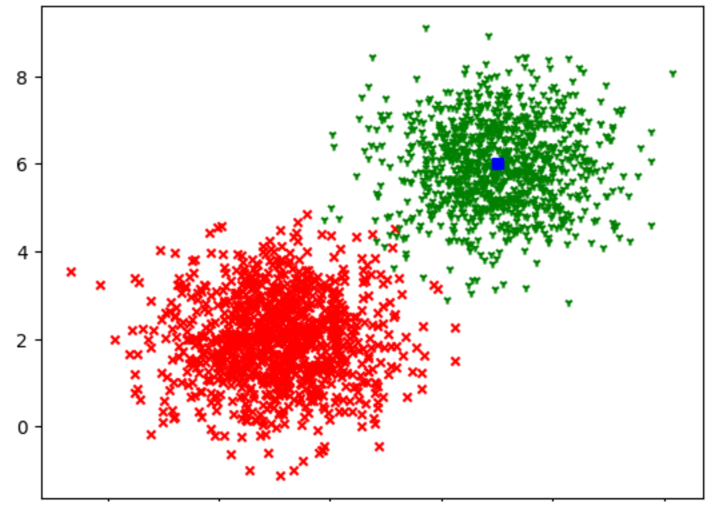


```
0.45143604
```

```
tf.sigmoid(a*x + b*y + c).numpy()
```

```
0.45143604
```

```
x=7.0
y=6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter(x,y,c='b', marker='s')
plt.show()
tf.sigmoid(a*x + b*y + c).numpy()
```




```
x=5.0
y=4.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
```