Import biblioteki **TensorFlow** (https://www.tensorflow.org/) z której będziemy korzystali w **uczeniu maszynowym**:

```
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
```

Dwa gangi

Przetesuj poniższe instrukcje:

```
[2]*12
```

```
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

```
[-3]*10+[4]*5
```

```
[-3, -3, -3, -3, -3, -3, -3, -3, -3, -3, 4, 4, 4, 4, 4]
```

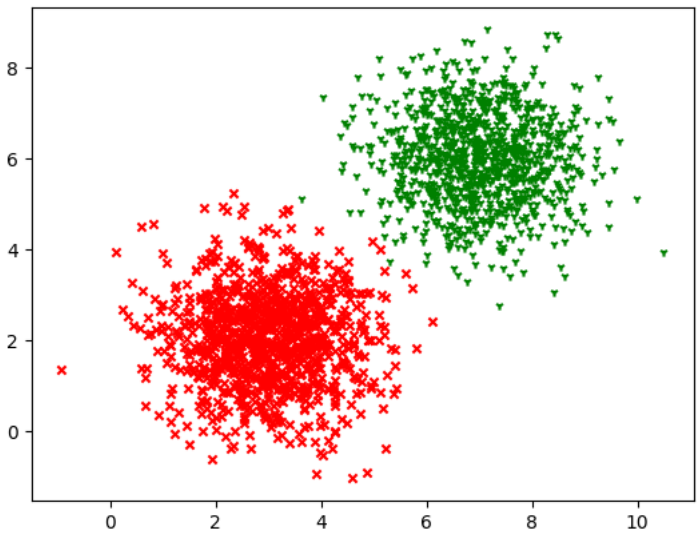
```
np.append([1,2,3],[4,5])
```

```
array([1, 2, 3, 4, 5])
```

Przygotowujemy 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) #tablica wsp. x dla 2000 punktów
ys = np.append(y_label1, y_label2) #tablica wsp. y dla 2000 punktów
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()
```



Wersja podstawowa

Definiujemy model:

```
model = Sequential()
```

Dodajemy **jedną warstwę** (Dense) z **jednym neuronem** (units=1) z **biasem** (use_bias=True) i **liniową funkcją aktywacji** (activation="linear"):

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

Definiujemy **optymalizator** i **błąd** (entropia krzyżowa). **Współczynnik uczenia = 0.1**

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)


model.compile(loss='binary_crossentropy',optimizer=opt)
```

Informacja o modelu:

```
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Przygotowanie danych:

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

Proces **uczenia**:

```
epochs = 100
h = model.fit(data_points,labels, verbose=1, epochs=epochs)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.0298
Epoch 81/100
63/63 [==============================] - 0s 3ms/step - loss: 0.0298
Epoch 82/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0296
Epoch 83/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0294
Epoch 84/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0291
Epoch 85/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0289
Epoch 86/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0286
Epoch 87/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0284
Epoch 88/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0283
Epoch 89/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0280
Epoch 90/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0279
Epoch 91/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0277
Epoch 92/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0274
Epoch 93/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0272
Epoch 94/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0270
Epoch 95/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0269
Epoch 96/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0267
Epoch 97/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0266
Epoch 98/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0263
Epoch 99/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0262
Epoch 100/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0260
```

```
Loss = h.history['loss']
Loss
```

```
 0.04586770758032799,
 0.04512612521648407,
 0.04437927529215813,
```

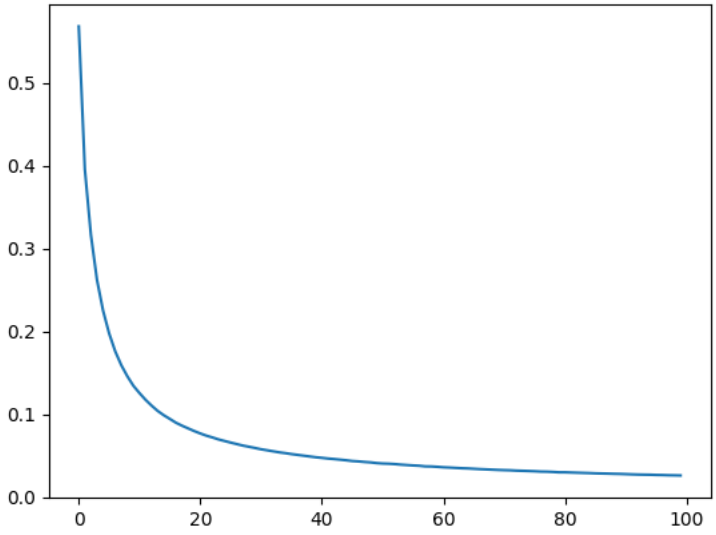
```
0.02826798334717505,
0.02804393880069256,
0.027902692556381226,
0.02765718288719654,
0.027381207793951035,
0.027161255478858948,
0.027026411145925522,
0.02693208120763302,
0.026673540472984314,
0.026553962379693985,
0.026313211768865585,
0.026199981570243835,
0.026036502793431282]
```

Sprawdźmy jakie są **wartości wag**:

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[1.1798693]
 [1.3720845]]
[-11.223719]
```

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



Sprawdzamy działanie modelu dla punktu o współrzędnych **x** i **y**:

```
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()
model.predict([[x,y]])
```

Liczba epok 50

```
model = Sequential()
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

```
epochs = 50
h = model.fit(data_points,labels, verbose=1, epochs=epochs)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.0575
Epoch 32/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0562
Epoch 33/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0548
Epoch 34/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0538
Epoch 35/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0528
Epoch 36/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0517
Epoch 37/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0507
Epoch 38/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0499
Epoch 39/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0488
Epoch 40/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0480
Epoch 41/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0472
Epoch 42/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0463
Epoch 43/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0456
Epoch 44/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0448
Epoch 45/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0441
Epoch 46/50
63/63 [==============================] - 0s 1ms/step - loss: 0.0435
Epoch 47/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0430
Epoch 48/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0423
Epoch 49/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0415
Epoch 50/50
63/63 [==============================] - 0s 2ms/step - loss: 0.0411
```

```
Loss = h.history['loss']
Loss
```

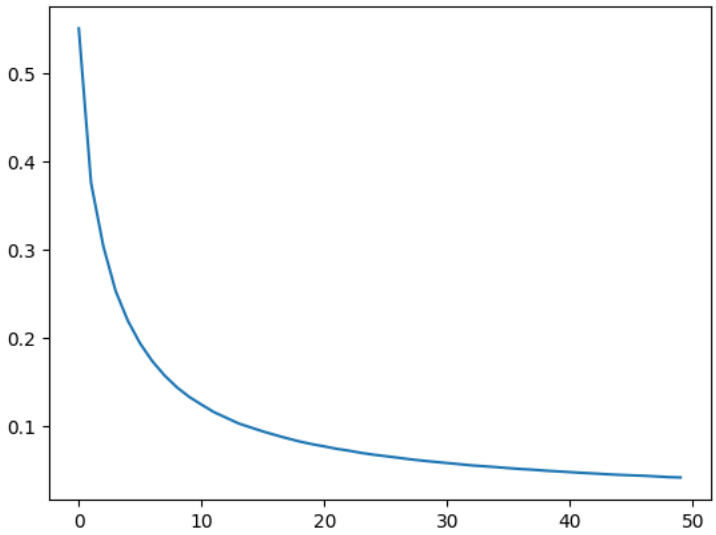
```
[0.5514559745788574,
 0.37589436769485474,
 0.30392253398895264,
 0.25305691361427307,
 0.21908478438854218,
 0.1932816207408905,
 0.17304490506649017,
 0.15681524574756622,
 0.14344017207622528,
 0.13260763883590698,
 0.12385395169258118,
 0.11542092263698578,
 0.10901965200901031,
 0.10257185250520706,
 0.09792258590459824,
 0.09340578317642212,
 0.08933097869157791,
 0.08550417423248291,
 0.08192317932844162,
 0.07893263548612595,
 0.07638262957334518,
 0.07367561012506485,
 0.07151411473751068,
 0.06906156986951828,
 0.06696797162294388,
 0.06528440117835999,
 0.06353318691253662,
 0.061711035668849945,
 0.060197532176971436,
 0.05881044268608093,
 0.05751041695475578,
 0.05620032548904419,
 0.05482563376426697,
 0.05382918938994408,
 0.052831731736660004,
 0.051722679287195206,
 0.05071362853050232,
 0.04988562688231468,
 0.04881628602743149,
 0.04801720753312111,
 0.04717530682682991,
 0.04633312299847603,
 0.045607224106788635,
 0.044752947986125946,
 0.04406304284930229,
 0.043541599065065384,
 0.04303644597530365,
```

```
       0.04232398420572281,
       0.04150399938225746,
       0.04108329862356186]
```

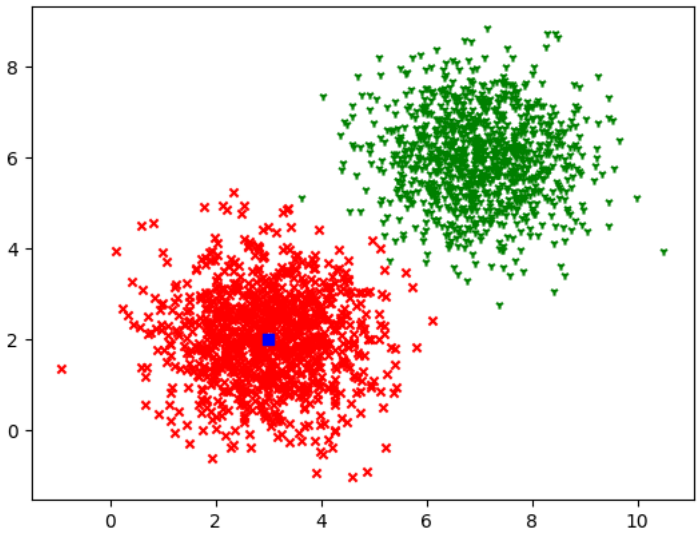
```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
     [[0.9366239]
      [1.1662605]]
     [-9.12026]
```

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




```
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()
model.predict([[x,y]])
```




```
1/1 [==============================] - 0s 57ms/step
array([[0.01838133]], dtype=float32)
```

Liczba epok 150

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)


model.compile(loss='binary_crossentropy',optimizer=opt)


model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

```
epochs = 150
h = model.fit(data_points,labels, verbose=1, epochs=epochs)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.0211
Epoch 141/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0210
Epoch 142/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0209
Epoch 143/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0209
Epoch 144/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0207
Epoch 145/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0206
Epoch 146/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0206
Epoch 147/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0204
Epoch 148/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0203
Epoch 149/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0203
Epoch 150/150
63/63 [==============================] - 0s 2ms/step - loss: 0.0203
```

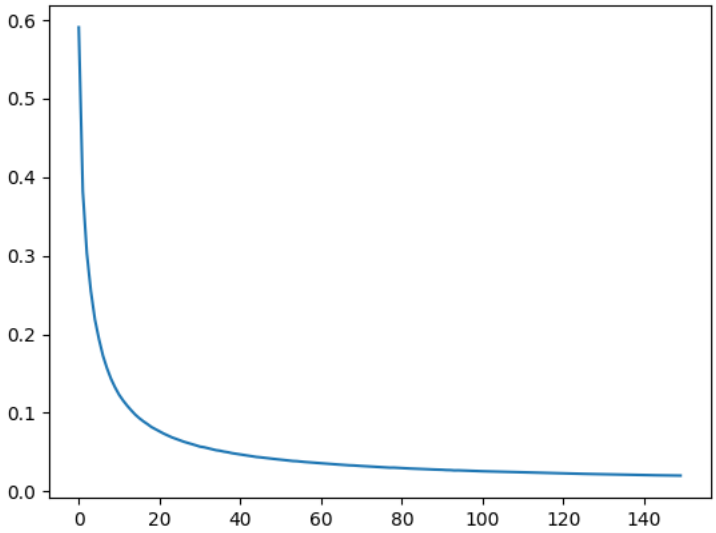
```
Loss = h.history['loss']
Loss
```

```
 0.027225833386182785,
 0.0268409326672554,
 0.026904912665486336,
 0.026697086170315742,
 0.026432672515511513,
 0.026267772540450096,
 0.026058020070195198,
 0.025939948856830597,
 0.025791969150304794,
 0.025651905685663223,
 0.025492770597338676,
 0.0253530815243721,
 0.025220762938261032,
 0.025000987574458122,
 0.024954764172434807,
 0.02475311979651451,
 0.024578964337706566,
 0.024441009387373924,
 0.02437330223619938,
 0.024173716083168983,
 0.023982660844922066,
 0.024098223075270653,
 0.02375289611518383,
 0.02372477948665619,
 0.023587016388773918,
 0.02339467778801918,
 0.023326121270656586,
 0.02320791967213154,
 0.023054776713252068,
 0.022858936339616776,
 0.022813567891716957,
 0.022726956754922867,
 0.022591205313801765,
 0.022460609674453735,
 0.022436462342739105,
 0.02224821411073208,
 0.0221633929759264,
 0.021990269422531128,
 0.021996568888425827,
 0.021830694749951363,
 0.02169998362660408,
 0.021673738956451416,
 0.02162276767194271,
 0.02152191661298275,
 0.021364254876971245,
 0.021210771054029465,
 0.0211164690554142,
 0.02105586603283882,
 0.02100347727537155,
 0.02090325579047203,
 0.020871158689260483,
 0.0206666961312294,
 0.020589370280504227,
 0.020557334646582603,
 0.02041422389447689,
 0.020283309742808342,
 0.020280789583921432,
 0.020253213122487068]
```

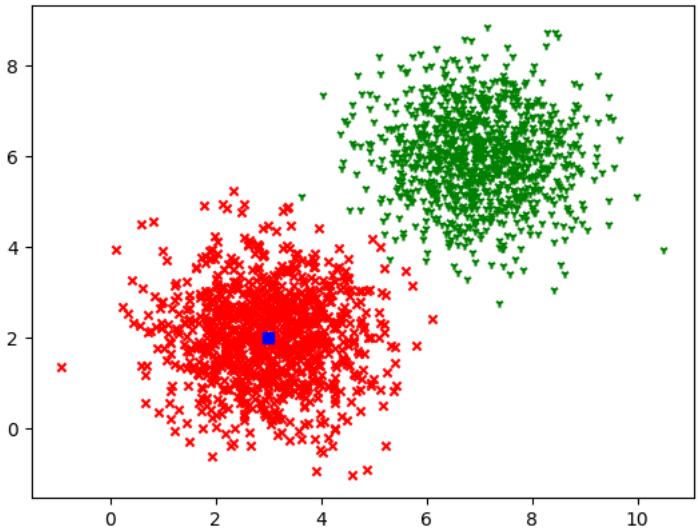
```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[1.3322845]
 [1.5147399]]
[-12.54979]
```

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




```
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()
model.predict([[x,y]])
```




```
1/1 [==============================] - 0s 57ms/step
array([[0.00397615]], dtype=float32)
```

współczynnik uczenia 0.01 (SGD)

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.01)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_13 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

```
epochs = 100
h = model.fit(data_points,labels, verbose=1, epochs=epochs)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.1334
Epoch 95/100
63/63 [==============================] - 0s 2ms/step - loss: 0.1323
Epoch 96/100
63/63 [==============================] - 0s 2ms/step - loss: 0.1314
Epoch 97/100
63/63 [==============================] - 0s 2ms/step - loss: 0.1304
Epoch 98/100
63/63 [==============================] - 0s 2ms/step - loss: 0.1295
Epoch 99/100
63/63 [==============================] - 0s 2ms/step - loss: 0.1284
Epoch 100/100
63/63 [==============================] - 0s 2ms/step - loss: 0.1276
```

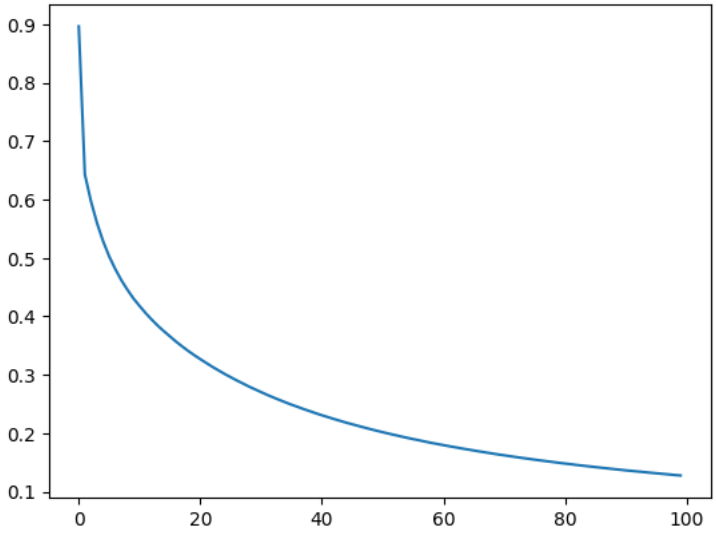
```
Loss = h.history['loss']
Loss
```

```
[0.8962318897247314,
 0.6430747509002686,
 0.5972104072570801,
 0.5595943927764893,
 0.5286520719528198,
 0.5026587843894958,
 0.4810537099838257,
 0.46194374561309814,
 0.4455479681491852,
 0.43051889538764954,
 0.4177596867084503,
 0.4055574834346771,
 0.39442387223243713,
 0.3839835822582245,
 0.3745209574699402,
 0.3657049238681793,
 0.35670554637908936,
 0.3485909700393677,
 0.340772807598114,
 0.3335612118244171,
 0.3266458511352539,
 0.3200240731239319,
 0.3134460151195526,
 0.3074128031730652,
 0.3013570010662079,
 0.29578179121017456,
 0.29021796584129333,
 0.28510165214538574,
 0.27976781129837036,
 0.27499574422836304,
 0.27027931809425354,
 0.26559650897979736,
 0.26119476556777954,
 0.2569583058357239,
 0.2527886927127838,
 0.24871297180652618,
 0.24505950510501862,
 0.2411756068468094,
 0.23771420121192932,
 0.23403947055339813,
 0.23067164421081543,
 0.22743365168571472,
 0.22421933710575104,
 0.22108790278434753,
 0.21801164746284485,
 0.21521519124507904,
 0.21232736110687256,
 0.20960623025894165,
 0.2068055421113968,
 0.2042142152786255,
 0.20166869461536407,
 0.19918698072433472,
 0.19685515761375427,
 0.19443300366401672,
 0.19209223985671997,
 0.18988706171512604,
 0.18776972591876984,
 0.18553286790847778,
```

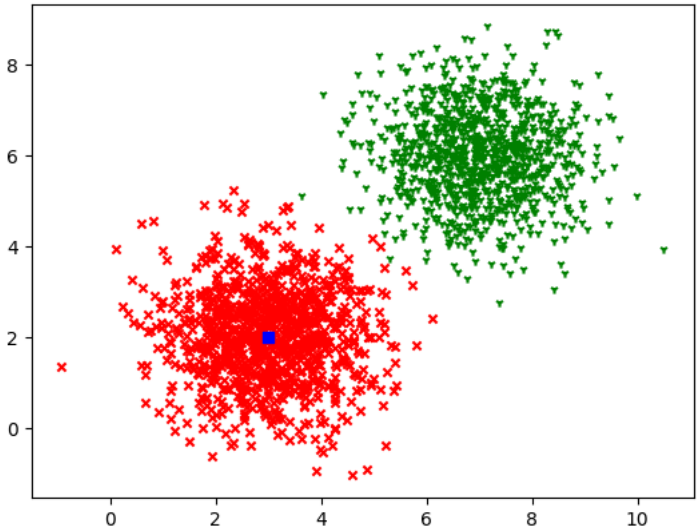
```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[0.42270604]
 [0.772689  ]]
[-4.849731]
```

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




```
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()
model.predict([[x,y]])
```




```
1/1 [==============================] - 0s 59ms/step
array([[0.11545069]], dtype=float32)
```

współczynnik uczenia 0.01 (Adam)

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
opt = tf.keras.optimizers.Adam(learning_rate=0.1)
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_14 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
```

```
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

```
epochs = 100
h = model.fit(data_points,labels, verbose=1, epochs=epochs)
```

```
63/63 [==============================] - 0s 2ms/step - loss: 0.0050
Epoch 100/100
63/63 [==============================] - 0s 2ms/step - loss: 0.0045
```

```
Loss = h.history['loss']
Loss
```

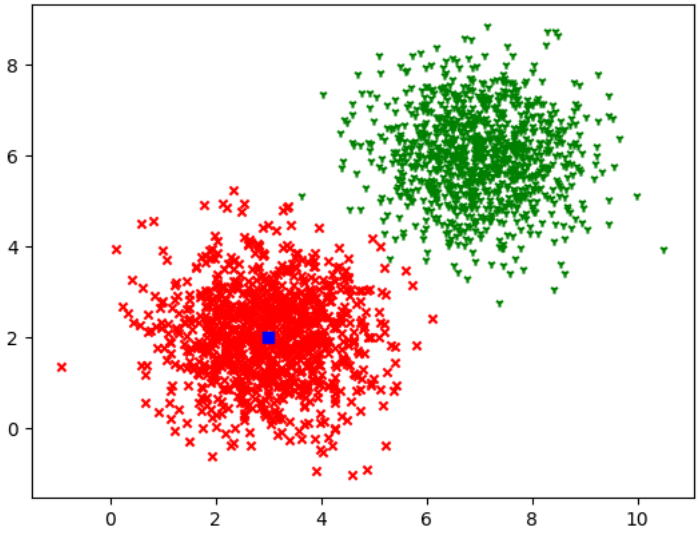
```
 0.007610319182276726,
 0.007542072795331478,
 0.007214203476905823,
 0.007276782765984535,
 0.006630009040236473,
 0.006203667260706425,
 0.00830827932804823,
 0.006748811807483435,
 0.006087803281843662,
 0.00844080098807812,
 0.006225721444934606,
 0.00592472730204463,
 0.005932462401688099,
 0.006011147052049637,
 0.005896668881177902,
 0.005850079469382763,
 0.007423819042742252,
 0.006593319587409496,
 0.005880228243768215,
 0.005982018541544676,
 0.005214039236307144,
 0.005543160252273083,
 0.006139489356428385,
 0.005309165455400944,
 0.006015453487634659,
 0.0066652242094278336,
 0.004830997437238693,
 0.005272964481264353,
 0.005532494280487299,
 0.005322652868926525,
 0.006496994756162167,
 0.005781716201454401,
 0.005802735686302185,
 0.005304526537656784,
 0.004926038905978203,
 0.005238265264779329,
 0.0044909375719726086,
 0.006666775792837143,
 0.0045027537271380424,
 0.006166620180010796,
 0.006509328726679087,
 0.004363068379461765,
 0.004996995907276869,
 0.00490476144477725,
 0.005580027122050524,
 0.004867125302553177,
 0.004148117266595364,
 0.0042757391929626465,
 0.0067716012708842754,
 0.004818758927285671,
 0.0050217523239552975,
 0.006061570253223181,
 0.0053549021050334,
 0.004575807135552168,
 0.009330855682492256,
 0.004141620360314846,
 0.004958016332238913,
 0.004460631404072046]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[3.2493508]
 [4.0460324]]
[-32.107903]
```

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


```
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()
model.predict([[x,y]])
```




```
1/1 [==============================] - 0s 58ms/step
array([[6.361797e-07]], dtype=float32)
```

Batch 100

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
```

```
            [3.45971985],
            [2.71106667],
            [3.16673227],
            [4.60943899],
            [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

```
epochs = 100
h = model.fit(data_points,labels, verbose=1, epochs=epochs,batch_size=100)
```

```
Epoch 72/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0714
Epoch 73/100
20/20 [==============================] - 0s 2ms/step - loss: 0.0706
Epoch 74/100
20/20 [==============================] - 0s 2ms/step - loss: 0.0701
Epoch 75/100
20/20 [==============================] - 0s 2ms/step - loss: 0.0693
Epoch 76/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0685
Epoch 77/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0680
Epoch 78/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0673
Epoch 79/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0667
Epoch 80/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0661
Epoch 81/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0655
Epoch 82/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0649
Epoch 83/100
20/20 [==============================] - 0s 2ms/step - loss: 0.0643
Epoch 84/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0638
Epoch 85/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0632
Epoch 86/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0628
Epoch 87/100
20/20 [==============================] - 0s 4ms/step - loss: 0.0622
Epoch 88/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0617
Epoch 89/100
20/20 [==============================] - 0s 2ms/step - loss: 0.0612
Epoch 90/100
20/20 [==============================] - 0s 2ms/step - loss: 0.0607
Epoch 91/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0603
Epoch 92/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0599
Epoch 93/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0594
Epoch 94/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0589
Epoch 95/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0584
Epoch 96/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0579
Epoch 97/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0576
Epoch 98/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0572
Epoch 99/100
20/20 [==============================] - 0s 3ms/step - loss: 0.0566
Epoch 100/100
20/20 [==============================] - 0s 2ms/step - loss: 0.0563
```

```
Loss = h.history['loss']
Loss
```

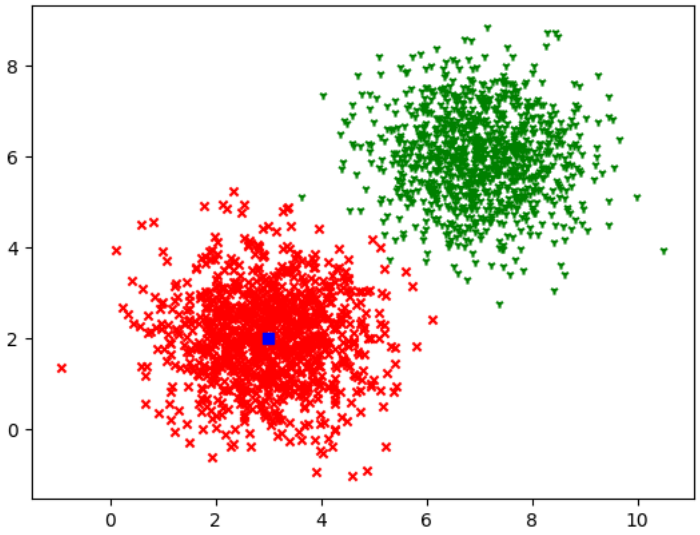
```
        0.09917806833982468,
        0.09765052050352097,
        0.0961403101682663,
        0.09469836205244064,
        0.09341509640216827,
        0.0918770432472229,
        0.09056208282709122,
        0.0894060879945755,
        0.08819734305143356,
        0.08696483820676804,
        0.08582387119531631,
        0.08465886861085892,
        0.08364276587963104,
        0.08251994103193283,
        0.08157859742641449,
        0.08054795116186142,
        0.07963430881500244,
        0.07877790927886963,
        0.07783127576112747,
        0.0767928957939148,
        0.07602166384458542,
        0.07524653524160385,
        0.07437698543071747,
        0.07370986044406891,
        0.07286348193883896,
        0.0719030499458313,
        0.0713948979973793,
        0.07060807198286057,
        0.07011864334344864,
        0.06927027553319931,
        0.0685458555817604,
        0.06797070801258087,
        0.06731429696083069,
        0.06670926511287689,
        0.06605261564254761,
        0.06549173593521118,
        0.06492602080106735,
        0.06429897993803024,
        0.06380020827054977,
        0.06324801594018936,
        0.06276465952396393,
        0.062200672924518585,
        0.06167246028780937,
        0.06121930107474327,
        0.06070210784673691,
        0.06026317551732063,
        0.059882164001464844,
        0.05935443937778473,
        0.05891241878271103,
        0.058400098234415054,
        0.05794026330113411,
        0.057552583515644073,
        0.05715750530362129,
        0.056646741926670074,
        0.05625966563820839]
```

```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
     [[0.7890133]
      [1.0396947]]
     [-7.7969418]
```

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

```
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()
model.predict([[x,y]])
```




```
1/1 [==============================] - 0s 66ms/step
array([[0.03387839]], dtype=float32)
```

Batch 200

```
model = Sequential()

model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))

#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)

model.compile(loss='binary_crossentropy',optimizer=opt)

model.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

```
epochs = 100
h = model.fit(data_points,labels, verbose=1, epochs=epochs,batch_size=200)
```

```
Epoch 72/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1182
Epoch 73/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1169
Epoch 74/100
10/10 [==============================] - 0s 3ms/step - loss: 0.1158
Epoch 75/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1147
Epoch 76/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1136
Epoch 77/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1124
Epoch 78/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1115
Epoch 79/100
10/10 [==============================] - 0s 3ms/step - loss: 0.1103
Epoch 80/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1093
Epoch 81/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1084
Epoch 82/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1074
Epoch 83/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1065
Epoch 84/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1055
Epoch 85/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1046
Epoch 86/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1038
Epoch 87/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1028
Epoch 88/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1020
Epoch 89/100
10/10 [==============================] - 0s 2ms/step - loss: 0.1013
Epoch 90/100
10/10 [==============================] - 0s 3ms/step - loss: 0.1005
Epoch 91/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0995
Epoch 92/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0987
Epoch 93/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0979
Epoch 94/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0971
Epoch 95/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0964
Epoch 96/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0957
Epoch 97/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0949
Epoch 98/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0943
Epoch 99/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0935
Epoch 100/100
10/10 [==============================] - 0s 2ms/step - loss: 0.0928
```

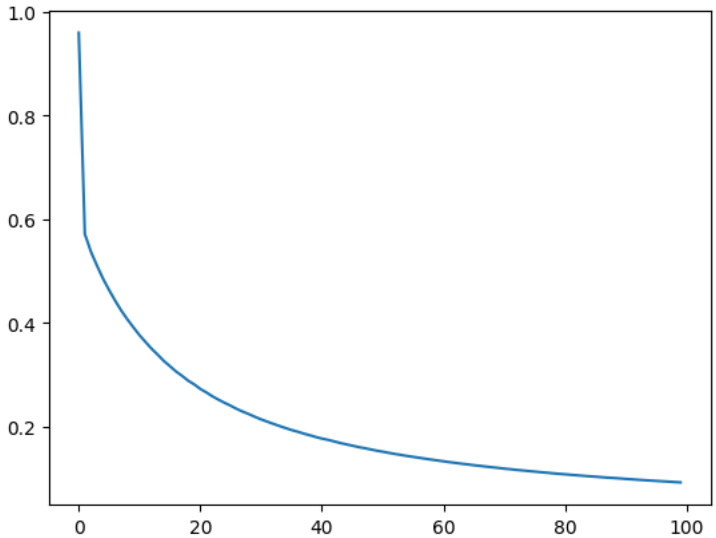
```
Loss = h.history['loss']
Loss
```

```
0.14766731858253481,
0.14574679732322693,
0.14376044273376465,
0.14199167490005493,
0.14024919271469116,
0.13843998312950134,
0.13675348460674286,
0.13509371876716614,
0.1334577053785324,
0.13177739083766937,
0.13033932447433472,
0.12876223027706146,
0.12743958830833435,
0.1258663386106491,
0.1246783584356308,
0.12324405461549759,
0.12195970118045807,
0.12071296572685242,
0.11931361258029938,
0.11822164803743362,
0.11694016307592392,
0.11584462225437164,
0.11465030908584595,
0.1136118546128273,
0.11243191361427307,
0.11149502545595169,
0.11033408343791962,
0.1092938706278801,
0.10840792953968048,
0.10737728327512741,
0.10647524893283844,
0.10548985004425049,
0.10461077094078064,
0.10378243774175644,
0.10282742977142334,
0.10198696702718735,
0.10129435360431671,
0.10045459866523743,
0.099464550614357,
0.09871871769428253,
0.09787728637456894,
0.09708764404058456,
0.09638604521751404,
0.09565816074609756,
0.09493104368448257,
0.0942513719201088,
0.09349773079156876,
0.09278937429189682]
```

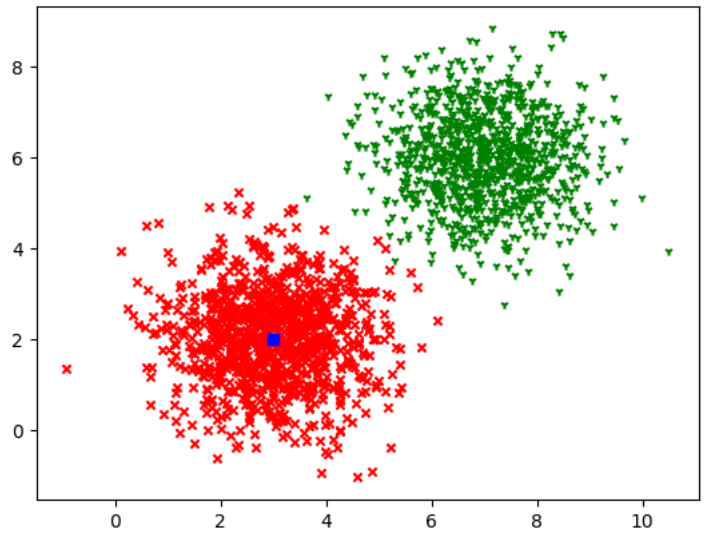
```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[0.5618733]
 [0.8686557]]
[-5.940121]
```

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



```
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()
model.predict([[x,y]])
```




```
1/1 [==============================] - 0s 98ms/step
array([[0.07466185]], dtype=float32)
```

Batch 400

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_17 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
```

```
            ...,
            [7.30751572, 5.85924988],
            [6.92047086, 6.74493881],
            [7.56260447, 4.96002013]])
```

```
epochs = 100
h = model.fit(data_points,labels, verbose=1, epochs=epochs,batch_size=400)
```

```
Epoch 72/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1930
Epoch 73/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1910
Epoch 74/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1894
Epoch 75/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1875
Epoch 76/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1860
Epoch 77/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1842
Epoch 78/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1829
Epoch 79/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1809
Epoch 80/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1793
Epoch 81/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1781
Epoch 82/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1764
Epoch 83/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1750
Epoch 84/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1735
Epoch 85/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1721
Epoch 86/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1706
Epoch 87/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1692
Epoch 88/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1679
Epoch 89/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1666
Epoch 90/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1652
Epoch 91/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1640
Epoch 92/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1627
Epoch 93/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1614
Epoch 94/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1602
Epoch 95/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1593
Epoch 96/100
5/5 [==============================] - 0s 4ms/step - loss: 0.1579
Epoch 97/100
5/5 [==============================] - 0s 5ms/step - loss: 0.1568
Epoch 98/100
5/5 [==============================] - 0s 3ms/step - loss: 0.1556
Epoch 99/100
5/5 [==============================] - 0s 5ms/step - loss: 0.1544
Epoch 100/100
5/5 [==============================] - 0s 5ms/step - loss: 0.1535
```

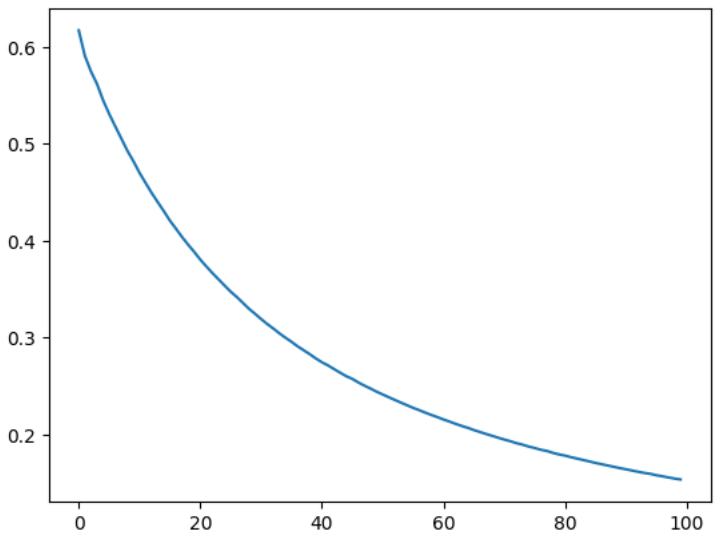
```
Loss = h.history['loss']
Loss
```

```
        0.21542641520500183,
        0.21321865916252136,
        0.21090559661388397,
        0.2087421417236328,
        0.2068212777376175,
        0.20448990166187286,
        0.20246070623397827,
        0.20046648383140564,
        0.19855184853076935,
        0.19658263027668,
        0.1946699172258377,
        0.1929682046175003,
        0.19098971784114838,
        0.18943087756633759,
        0.18750126659870148,
        0.18596020340919495,
        0.18417124450206757,
        0.1829407811164856,
        0.1809232532978058,
        0.17934291064739227,
        0.17810387909412384,
        0.17643415927886963,
        0.17503686249256134,
        0.17354489862918854,
        0.17210248112678528,
        0.17057481408119202,
        0.16924619674682617,
        0.16790054738521576,
        0.16658128798007965,
        0.16521747410297394,
        0.16402742266654968,
        0.16270218789577484,
        0.1614331752061844,
        0.16024380922317505,
        0.15932266414165497,
        0.15785802900791168,
        0.15675802528858185,
        0.1555551439523697,
        0.15442602336406708,
        0.15345922112464905]
```

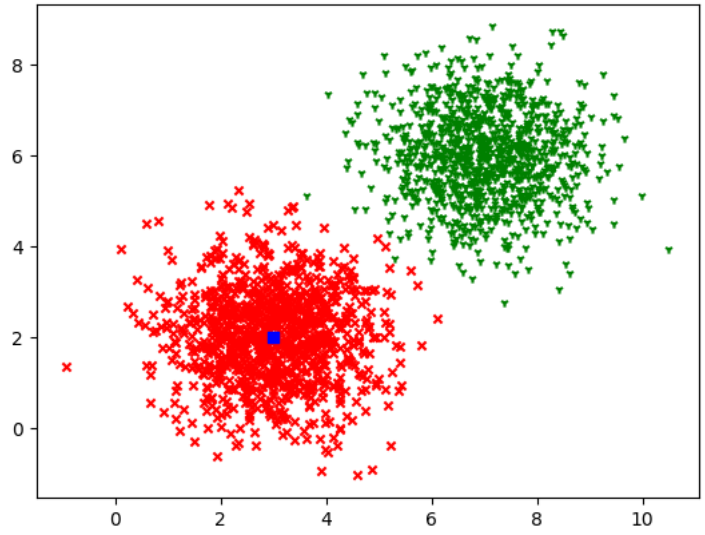
```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[0.3480096]
 [0.7281162]]
[-4.24621]
```

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




```
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()
model.predict([[x,y]])
```




```
1/1 [==============================] - 0s 92ms/step
```

Najlepszy model

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
opt = tf.keras.optimizers.Adam(learning_rate=0.1)
#opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
```

```
model.summary()
```

```
Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_20 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
xs[0:10].reshape(-1,1)
```

```
array([[3.6552444 ],
       [4.48335868],
       [3.16803826],
       [2.83961866],
       [1.8784655 ],
       [3.45971985],
       [2.71106667],
       [3.16673227],
       [4.60943899],
       [4.46992573]])
```

```
xs=xs.reshape(-1,1)
ys=ys.reshape(-1,1)
data_points=np.concatenate([xs,ys],axis=1)
data_points
```

```
array([[3.6552444 , 1.76714728],
       [4.48335868, 3.05201951],
       [3.16803826, 2.91245684],
       ...,
       [7.30751572, 5.85924988],
       [6.92047086, 6.74493881],
       [7.56260447, 4.96002013]])
```

```
epochs = 150
h = model.fit(data_points,labels, verbose=1, epochs=epochs,batch_size=150)
```

```
Epoch 122/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0090
Epoch 123/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0090
Epoch 124/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0088
Epoch 125/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0090
Epoch 126/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0091
Epoch 127/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0087
Epoch 128/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0086
Epoch 129/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0084
Epoch 130/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0085
Epoch 131/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0086
Epoch 132/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0084
Epoch 133/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0083
Epoch 134/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0082
Epoch 135/150
14/14 [==============================] - 0s 3ms/step - loss: 0.0082
Epoch 136/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0084
Epoch 137/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0082
Epoch 138/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0084
Epoch 139/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0089
Epoch 140/150
14/14 [==============================] - 0s 3ms/step - loss: 0.0083
Epoch 141/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0079
Epoch 142/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0078
Epoch 143/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0078
Epoch 144/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0081
Epoch 145/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0081
Epoch 146/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0081
Epoch 147/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0080
Epoch 148/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0075
Epoch 149/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0075
Epoch 150/150
14/14 [==============================] - 0s 2ms/step - loss: 0.0078
```

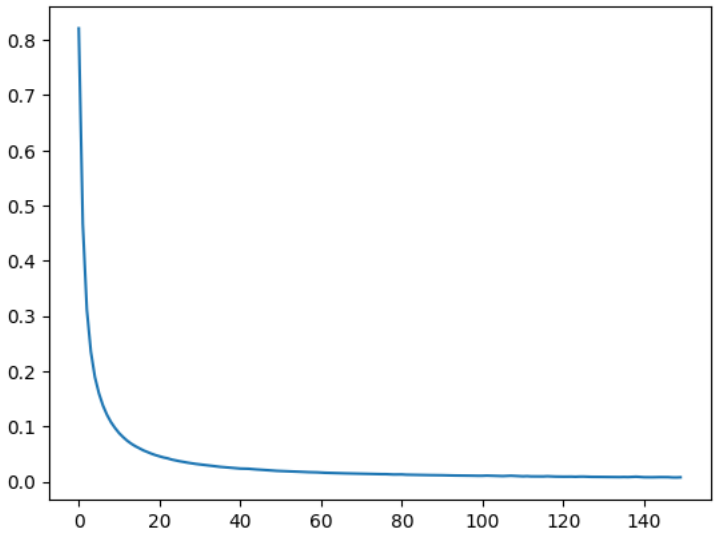
```
Loss = h.history['loss']
Loss
```

```
        0.008935256861150265,
        0.008921898901462555,
        0.00903440639376640 3,
        0.008985971100628376,
        0.008751117624342442,
        0.009043822064995766,
        0.009061303921043873,
        0.008735472336411476,
        0.008584128692746162,
        0.008431325666606426,
        0.008505510166287422,
        0.00856351014226675,
        0.008362445048987865,
        0.008259194903075695,
        0.008215419016778469,
        0.008175410330295563,
        0.008371892385184765,
        0.008191799744963646,
        0.008399713784456253,
        0.00886130053550005,
        0.008292256854474545,
        0.007921857759356499,
        0.0078056855127215385,
        0.007835110649466515,
        0.008076337166130543,
        0.00813664123415947,
        0.008058221079409122,
        0.008038878440856934,
        0.00754231633618474,
        0.0074712964706122875,
        0.007770325988531113]
```

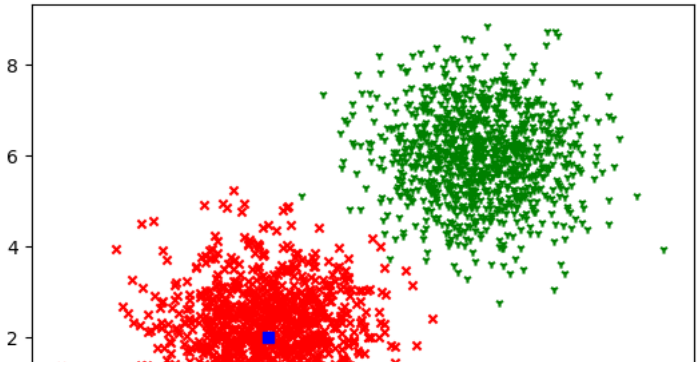
```
weights = model.get_weights()

print(weights[0])
print(weights[1])    #bias
```

```
[[2.0660715]
 [2.3702705]]
[-20.011349]
```

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




```
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()
model.predict([[x,y]])
```



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.

Wersja ze zbiorami treningowym i walidacyjnym

Wartości domyślne

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
#model.compile(loss='binary_crossentropy',optimizer=opt,metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_19 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 100
h = model.fit(data_points,labels,validation_split=0.2,verbose=1, epochs=epochs,batch_size=100)
```

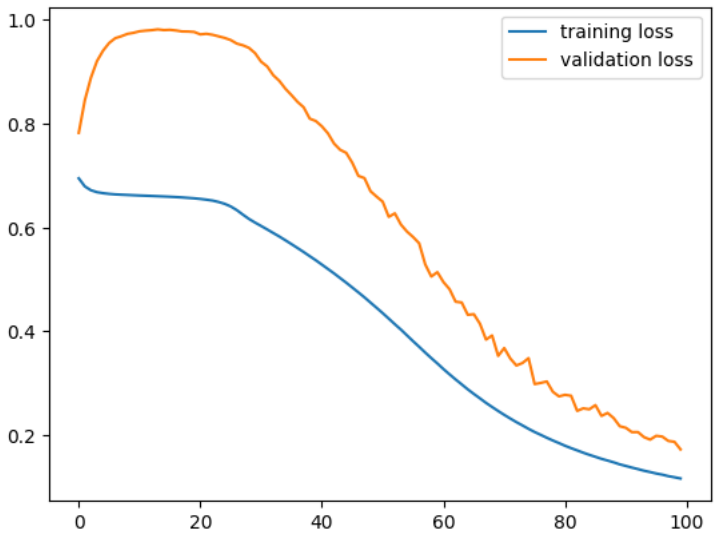
```
16/16 [==============================] - 0s 6ms/step - loss: 0.0996 - val_loss: 0.0883
Epoch 57/100
16/16 [==============================] - 0s 6ms/step - loss: 0.0982 - val_loss: 0.0752
Epoch 58/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0970 - val_loss: 0.0838
Epoch 59/100
16/16 [==============================] - 0s 6ms/step - loss: 0.0959 - val_loss: 0.0788
Epoch 60/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0946 - val_loss: 0.0735
Epoch 61/100
16/16 [==============================] - 0s 4ms/step - loss: 0.0934 - val_loss: 0.0811
Epoch 62/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0924 - val_loss: 0.0700
Epoch 63/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0915 - val_loss: 0.0733
Epoch 64/100
16/16 [==============================] - 0s 6ms/step - loss: 0.0905 - val_loss: 0.0762
Epoch 65/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0894 - val_loss: 0.0791
Epoch 66/100
16/16 [==============================] - 0s 4ms/step - loss: 0.0883 - val_loss: 0.0772
Epoch 67/100
16/16 [==============================] - 0s 6ms/step - loss: 0.0872 - val_loss: 0.0752
Epoch 68/100
16/16 [==============================] - 0s 6ms/step - loss: 0.0864 - val_loss: 0.0710
Epoch 69/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0854 - val_loss: 0.0679
Epoch 70/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0845 - val_loss: 0.0658
Epoch 71/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0837 - val_loss: 0.0667
Epoch 72/100
16/16 [==============================] - 0s 5ms/step - loss: 0.0827 - val_loss: 0.0796
Epoch 73/100
16/16 [==============================] - 0s 6ms/step - loss: 0.0822 - val_loss: 0.0718
Epoch 74/100
16/16 [==============================] - 0s 4ms/step - loss: 0.0812 - val_loss: 0.0672
Epoch 75/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0805 - val_loss: 0.0748
```

```
h.history.keys()
```

```
dict_keys(['loss', 'val_loss'])
```

```
Loss = h.history['loss']
Val_loss = h.history['val_loss']
```

```
plt.plot(Loss,label="training loss")
plt.plot(Val_loss,label="validation loss")
plt.legend()
plt.show()
```



Validation_split 0.6

```
model = Sequential()


model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))


#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)


model.compile(loss='binary_crossentropy',optimizer=opt)
#model.compile(loss='binary_crossentropy',optimizer=opt,metrics=['accuracy'])


model.summary()
```

```
Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_21 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 100
h = model.fit(data_points,labels,validation_split=0.6,verbose=1, epochs=epochs,batch_size=100)
```

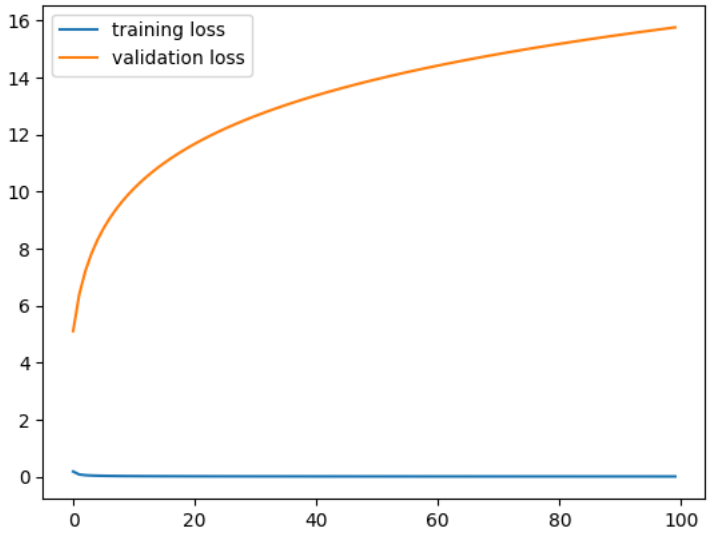
```
Epoch 72/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0034 - val_loss: 14.8632
Epoch 73/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0033 - val_loss: 14.9004
Epoch 74/100
8/8 [==============================] - 0s 18ms/step - loss: 0.0033 - val_loss: 14.9371
Epoch 75/100
8/8 [==============================] - 0s 18ms/step - loss: 0.0033 - val_loss: 14.9734
Epoch 76/100
8/8 [==============================] - 0s 17ms/step - loss: 0.0032 - val_loss: 15.0092
Epoch 77/100
8/8 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 15.0446
Epoch 78/100
8/8 [==============================] - 0s 17ms/step - loss: 0.0032 - val_loss: 15.0796
Epoch 79/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0031 - val_loss: 15.1141
Epoch 80/100
8/8 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss: 15.1483
Epoch 81/100
8/8 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss: 15.1820
Epoch 82/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0031 - val_loss: 15.2153
Epoch 83/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0030 - val_loss: 15.2483
Epoch 84/100
8/8 [==============================] - 0s 11ms/step - loss: 0.0030 - val_loss: 15.2809
Epoch 85/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0030 - val_loss: 15.3132
Epoch 86/100
8/8 [==============================] - 0s 10ms/step - loss: 0.0030 - val_loss: 15.3451
Epoch 87/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0029 - val_loss: 15.3766
Epoch 88/100
8/8 [==============================] - 0s 17ms/step - loss: 0.0029 - val_loss: 15.4078
Epoch 89/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0029 - val_loss: 15.4387
Epoch 90/100
8/8 [==============================] - 0s 10ms/step - loss: 0.0029 - val_loss: 15.4692
Epoch 91/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0028 - val_loss: 15.4995
Epoch 92/100
8/8 [==============================] - 0s 11ms/step - loss: 0.0028 - val_loss: 15.5294
Epoch 93/100
8/8 [==============================] - 0s 10ms/step - loss: 0.0028 - val_loss: 15.5590
Epoch 94/100
8/8 [==============================] - 0s 10ms/step - loss: 0.0028 - val_loss: 15.5884
Epoch 95/100
8/8 [==============================] - 0s 10ms/step - loss: 0.0027 - val_loss: 15.6174
Epoch 96/100
8/8 [==============================] - 0s 10ms/step - loss: 0.0027 - val_loss: 15.6462
Epoch 97/100
8/8 [==============================] - 0s 15ms/step - loss: 0.0027 - val_loss: 15.6746
Epoch 98/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0027 - val_loss: 15.7028
Epoch 99/100
8/8 [==============================] - 0s 18ms/step - loss: 0.0027 - val_loss: 15.7308
Epoch 100/100
8/8 [==============================] - 0s 16ms/step - loss: 0.0026 - val_loss: 15.7584
```

```
h.history.keys()
```

```
dict_keys(['loss', 'val_loss'])
```

```
Loss = h.history['loss']
Val_loss = h.history['val_loss']
```

```
plt.plot(Loss,label="training loss")
plt.plot(Val_loss,label="validation loss")
plt.legend()
plt.show()
```



Validation_split 0.1

```
model = Sequential()
```

```
model.add(Dense(units = 1, use_bias=True, input_dim=2, activation = "sigmoid"))
```

```
#opt = tf.keras.optimizers.Adam(learning_rate=0.1)
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

```
model.compile(loss='binary_crossentropy',optimizer=opt)
#model.compile(loss='binary_crossentropy',optimizer=opt,metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_22 (Dense)            (None, 1)                 3

=================================================================
Total params: 3 (12.00 Byte)
Trainable params: 3 (12.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
epochs = 100
h = model.fit(data_points,labels,validation_split=0.1,verbose=1, epochs=epochs,batch_size=100)
```

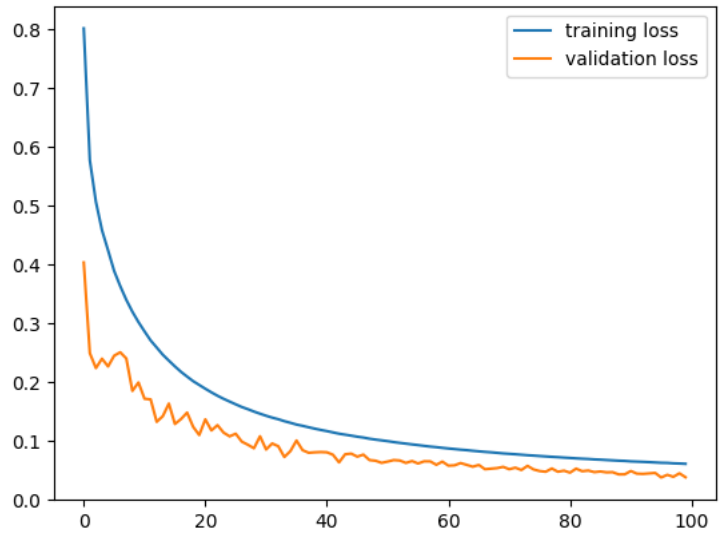
```
18/18 [==============================] - 0s 5ms/step - loss: 0.0740 - val_loss: 0.0484
Epoch 77/100
18/18 [==============================] - 0s 4ms/step - loss: 0.0733 - val_loss: 0.0473
Epoch 78/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0726 - val_loss: 0.0528
Epoch 79/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0720 - val_loss: 0.0472
Epoch 80/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0713 - val_loss: 0.0490
Epoch 81/100
18/18 [==============================] - 0s 4ms/step - loss: 0.0706 - val_loss: 0.0457
Epoch 82/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0700 - val_loss: 0.0525
Epoch 83/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0694 - val_loss: 0.0482
Epoch 84/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0689 - val_loss: 0.0493
Epoch 85/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0682 - val_loss: 0.0466
Epoch 86/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0678 - val_loss: 0.0476
Epoch 87/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0671 - val_loss: 0.0462
Epoch 88/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0666 - val_loss: 0.0465
Epoch 89/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0660 - val_loss: 0.0427
Epoch 90/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0656 - val_loss: 0.0428
Epoch 91/100
18/18 [==============================] - 0s 4ms/step - loss: 0.0649 - val_loss: 0.0485
Epoch 92/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0644 - val_loss: 0.0440
Epoch 93/100
18/18 [==============================] - 0s 5ms/step - loss: 0.0640 - val_loss: 0.0436
Epoch 94/100
18/18 [==============================] - 0s 4ms/step - loss: 0.0636 - val_loss: 0.0444
Epoch 95/100
18/18 [==============================] - 0s 4ms/step - loss: 0.0630 - val_loss: 0.0452
Epoch 96/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0625 - val_loss: 0.0375
Epoch 97/100
18/18 [==============================] - 0s 4ms/step - loss: 0.0623 - val_loss: 0.0420
Epoch 98/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0616 - val_loss: 0.0386
Epoch 99/100
18/18 [==============================] - 0s 3ms/step - loss: 0.0612 - val_loss: 0.0445
Epoch 100/100
18/18 [==============================] - 0s 4ms/step - loss: 0.0607 - val_loss: 0.0379
```

```
h.history.keys()
```

```
dict_keys(['loss', 'val_loss'])
```

```
Loss = h.history['loss']
Val_loss = h.history['val_loss']
```

```
plt.plot(Loss,label="training loss")
plt.plot(Val_loss,label="validation loss")
plt.legend()
plt.show()
```



Parametr validation split split mówi ile zbioru danych będzie przeznaczone na zbiór validacyjny a ile na zbiór treningowy. im większa wartość validation_split tym mniejszy jest zbiór treningowy. Powyższe wykresy pokazują, ze w wypadku użycia dużego zbioru validacyjnego(validation_split 0.6) model ma problemy się nauczyć.