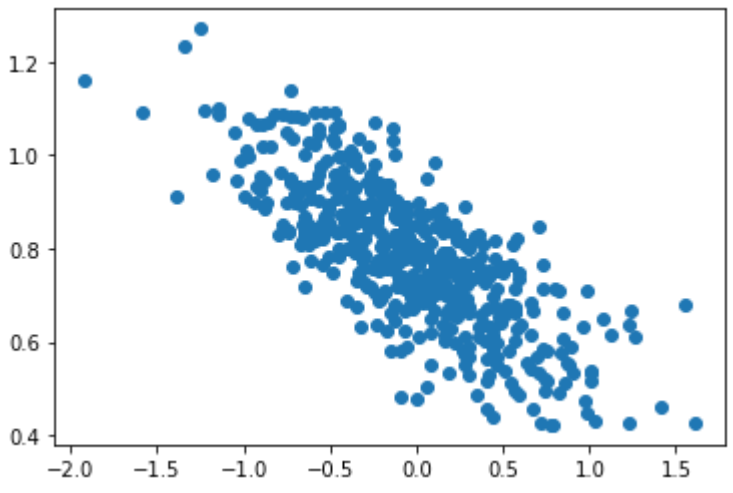
Teoria

[ ] ↳ *Скрыто 66 ячеек.*

Zadania

zad.1

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

number_of_points = 500
x_point = []
y_point = []

a = 0.22
b = 0.78

for i in range(number_of_points):
    x = np.random.normal(0.0,0.5)
    y = (-a*x+b)+np.random.normal(0.0,0.1)
    x_point.append(x)
    y_point.append(y)

plt.scatter(x_point,y_point,c='tab:blue')
plt.show()
```

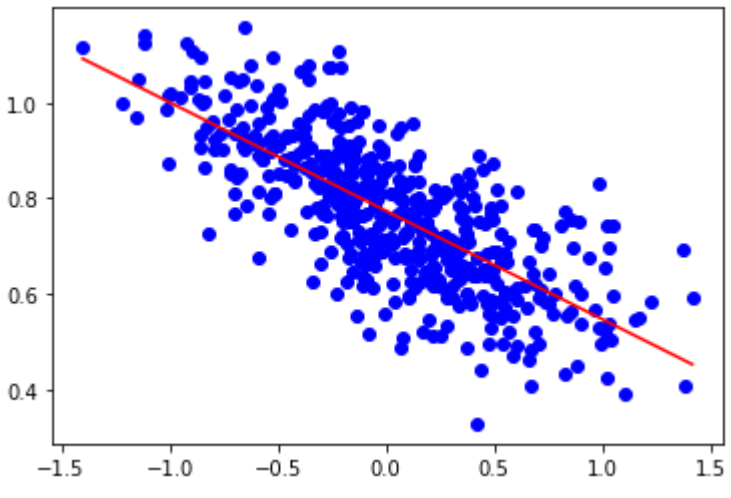



```
real_x = np.array(x_point)
real_y = np.array(y_point)

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

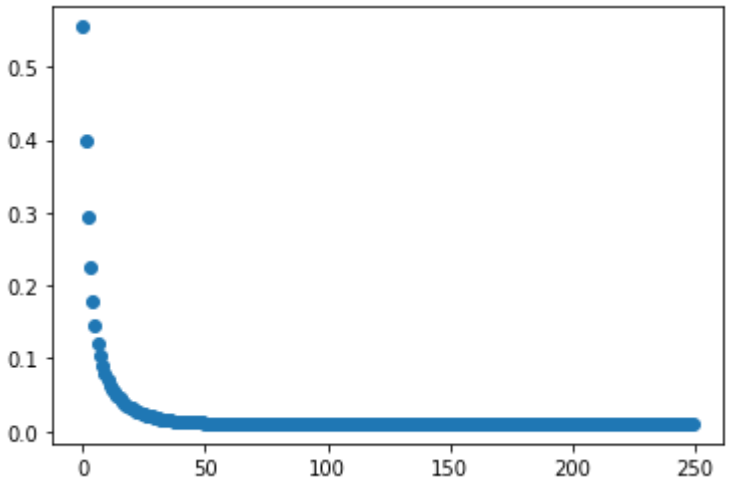
```
def loss_fn(real_y, pred_y):
  return tf.reduce_mean((real_y - pred_y)**2)


Loss = []
epochs = 250
learning_rate = 0.1

for _ in range(epochs):
  with tf.GradientTape() as tape:
    pred_y = a * real_x + b
    loss = loss_fn(real_y, pred_y)
    Loss.append(loss.numpy())

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

  a.assign_sub(learning_rate * dloss_da)
  b.assign_sub(learning_rate * dloss_db)

np.max(Loss), np.min(Loss)

print(a.numpy())
print(b.numpy())
```

```
-0.2272811
0.773717
```

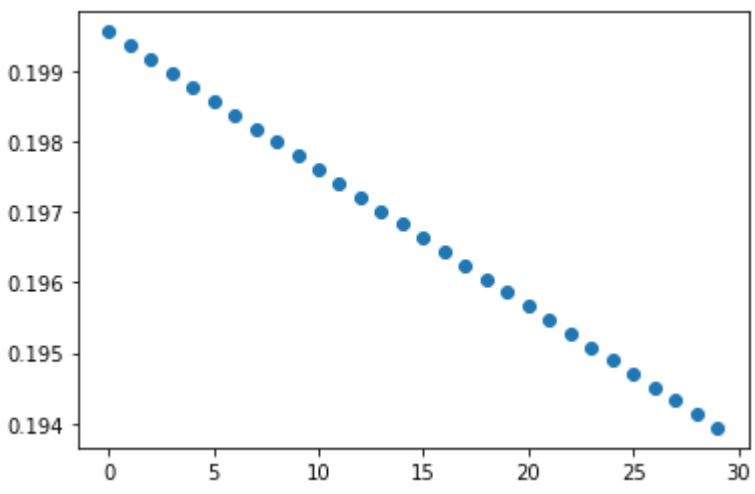
```
X = np.linspace(np.min(x_point), np.max(x_point), num = 10)

plt.plot(X,a.numpy()*X+b.numpy(),c='r')
plt.scatter(x_point,y_point,c="b")
plt.show()

# błąd wolniej się zmniejsza ze zmiejszeniem wartości uczenia
# mamy więcej wartości na wykresie ze wrzostem wartości "epochs"
```




```
plt.scatter(np.arange(epochs),Loss)
plt.show()
```

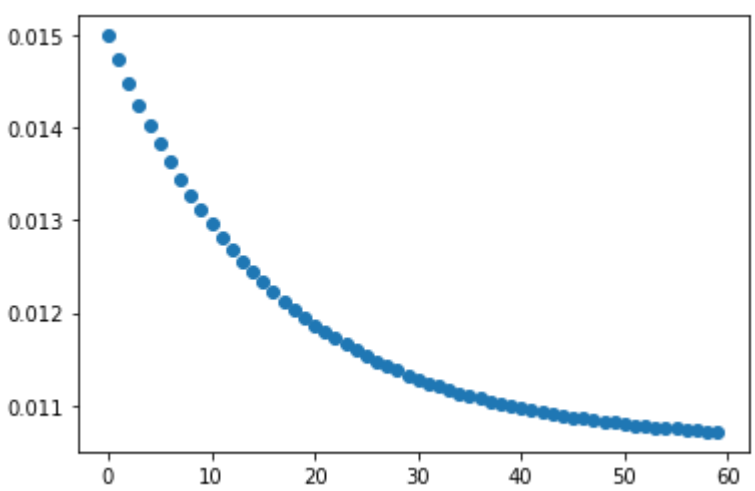



```
max = np.max(x_point)
min = np.min(x_point)


Loss = []
epochs = 30
learning_rate = 0.001

for _ in range(epochs):
  with tf.GradientTape() as tape:
    pred_y = a * real_x + b
    loss = loss_fn(real_y, pred_y)
    Loss.append(loss.numpy())

  grad_a, grad_b = tape.gradient(loss,(a, b))

  a.assign_sub(learning_rate*grad_a)
  b.assign_sub(learning_rate*grad_b)

plt.scatter(np.arange(epochs),Loss)
plt.show()
```

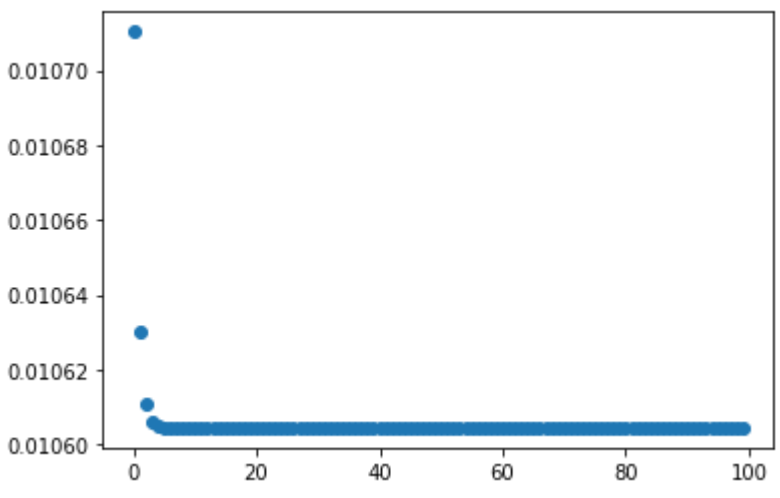



```
Loss = []
epochs = 60
learning_rate = 0.06

for _ in range(epochs):
```

```
  with tf.GradientTape() as tape:
    pred_y = a * real_x + b
    loss = loss_fn(real_y, pred_y)
    Loss.append(loss.numpy())

  grad_a, grad_b = tape.gradient(loss,(a, b))

  a.assign_sub(learning_rate*grad_a)
  b.assign_sub(learning_rate*grad_b)

plt.scatter(np.arange(epochs),Loss)
plt.show()
```




```
Loss = []
epochs = 100
learning_rate = 1.0


for _ in range(epochs):
  with tf.GradientTape() as tape:
    pred_y = a * real_x + b
    loss = loss_fn(real_y, pred_y)
    Loss.append(loss.numpy())

  grad_a, grad_b = tape.gradient(loss,(a, b))

  a.assign_sub(learning_rate*grad_a)
  b.assign_sub(learning_rate*grad_b)

plt.scatter(np.arange(epochs),Loss)
plt.show()
```

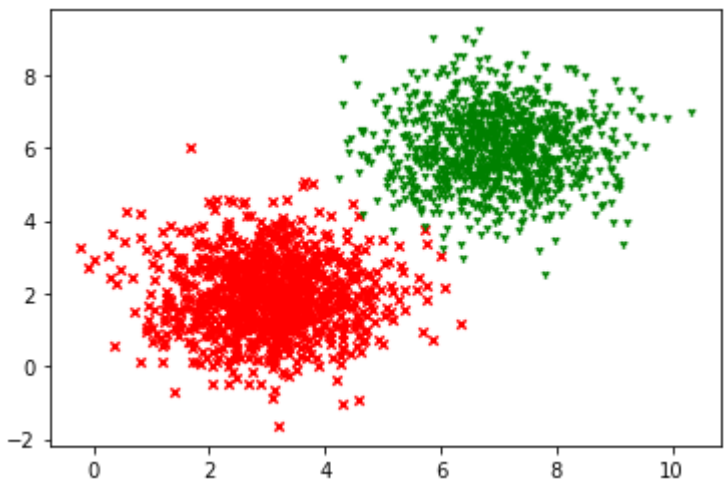



```
Loss = []
epochs = 500
learning_rate = 0.1


for _ in range(epochs):
  with tf.GradientTape() as tape:
    pred_y = a * real_x + b
    loss = loss_fn(real_y, pred_y)
    Loss.append(loss.numpy())

  grad_a, grad_b = tape.gradient(loss,(a, b))

  a.assign_sub(learning_rate*grad_a)
  b.assign_sub(learning_rate*grad_b)

plt.scatter(np.arange(epochs),Loss)
plt.show()
```

zad 2

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


x_label_1 = np.random.normal(3, 1, 1000)
y_label_1 = np.random.normal(2, 1, 1000)
x_label_2 = np.random.normal(7, 1, 1000)
y_label_2 = np.random.normal(6, 1, 1000)

xs = np.append(x_label_1, x_label_2)
ys = np.append(y_label_1, y_label_2)
labels = np.asarray([0.]*len(x_label_1) + [1.]*len(x_label_2))
```

```
plt.scatter(x_label_1, y_label_1, c='r', marker='x', s=20)
plt.scatter(x_label_2, y_label_2, c='g', marker='1', s=20)
plt.show()
```




```
x = np.vstack((xs, ys)).T
print(x)
print()
print(np.transpose(x))
```

```
[[3.48787903 3.24895974]
 [3.50626819 2.60484487]
 [1.99147658 0.85873913]
 ...
 [8.33599718 8.11762084]
 [6.52959012 6.34064173]
 [8.04083645 5.09289062]]

[[3.48787903 3.50626819 1.99147658 ... 8.33599718 6.52959012 8.04083645]
 [3.24895974 2.60484487 0.85873913 ... 8.11762084 6.34064173 5.09289062]]
```

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


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

real_x1 = np.array(x_label_1)
real_y1 = np.array(y_label_1)
real_x2 = np.array(x_label_2)
real_y2 = np.array(y_label_2)

Loss = []
epochs = 1000
learning_rate = 0.001


model = Sequential()
opt = keras.optimizers.SGD(learning_rate=learning_rate)
model.add(Dense(units = 1, use_bias = True, input_dim = 2, activation = "sigmoid"))
```

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

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

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

```
array([[4.23167882, 1.44365017],
       [3.79270108, 1.80743593],
       [3.05961926, 1.51418066],
       ...,
       [8.08599218, 4.21467478],
       [7.71952223, 5.82745873],
       [8.02318732, 6.77893381]])
```

```
h = model.fit(data_points, labels, verbose = 1, epochs = epochs, batch_size = 1000)
```

```
2/2 [==============================] - 0s 2ms/step - loss: 0.5127
Epoch 972/1000
2/2 [==============================] - 0s 3ms/step - loss: 0.5077
Epoch 973/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5140
Epoch 974/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5129
Epoch 975/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5086
Epoch 976/1000
2/2 [==============================] - 0s 3ms/step - loss: 0.5122
Epoch 977/1000
2/2 [==============================] - 0s 9ms/step - loss: 0.5175
Epoch 978/1000
2/2 [==============================] - 0s 8ms/step - loss: 0.5134
Epoch 979/1000
2/2 [==============================] - 0s 6ms/step - loss: 0.5124
Epoch 980/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5079
Epoch 981/1000
2/2 [==============================] - 0s 3ms/step - loss: 0.5094
Epoch 982/1000
2/2 [==============================] - 0s 3ms/step - loss: 0.5127
Epoch 983/1000
2/2 [==============================] - 0s 9ms/step - loss: 0.5141
Epoch 984/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5096
Epoch 985/1000
2/2 [==============================] - 0s 6ms/step - loss: 0.5166
Epoch 986/1000
2/2 [==============================] - 0s 9ms/step - loss: 0.5129
Epoch 987/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5078
Epoch 988/1000
2/2 [==============================] - 0s 6ms/step - loss: 0.5097
Epoch 989/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5169
Epoch 990/1000
2/2 [==============================] - 0s 2ms/step - loss: 0.5125
Epoch 991/1000
2/2 [==============================] - 0s 3ms/step - loss: 0.5134
Epoch 992/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5084
Epoch 993/1000
2/2 [==============================] - 0s 3ms/step - loss: 0.5089
Epoch 994/1000
2/2 [==============================] - 0s 6ms/step - loss: 0.5101
Epoch 995/1000
2/2 [==============================] - 0s 5ms/step - loss: 0.5160
Epoch 996/1000
2/2 [==============================] - 0s 6ms/step - loss: 0.5085
Epoch 997/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5129
Epoch 998/1000
2/2 [==============================] - 0s 3ms/step - loss: 0.5103
Epoch 999/1000
2/2 [==============================] - 0s 4ms/step - loss: 0.5055
Epoch 1000/1000
2/2 [==============================] - 0s 5ms/step - loss: 0.5100
```

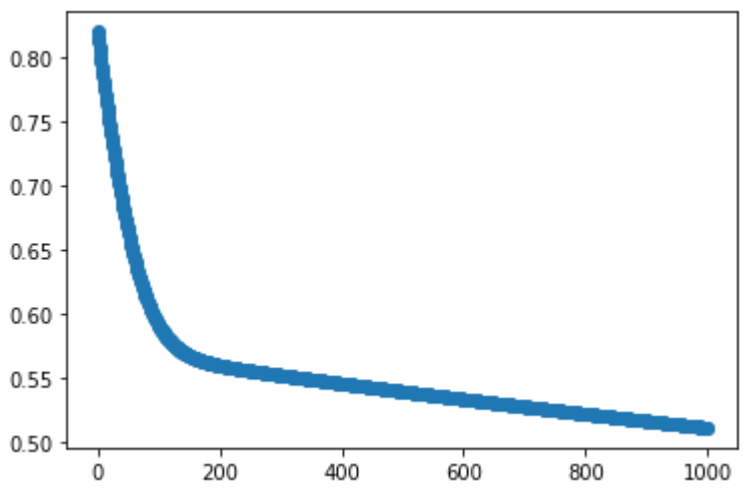
```
Loss = h.history['loss']
```

```
Loss
      0.5139447450637817,
      0.5138922333717346,
      0.5138370394706726,
      0.5137850046157837,
      0.5137293934822083,
      0.5136758089065552,
      0.513621985912323,
      0.5135681629180908,
      0.513514518737793,
      0.5134639739990234,
      0.5134080052375793,
      0.5133551955223083,
      0.5133007764816284,
      0.513246476650238,
      0.5131932497024536,
      0.5131410956382751,
      0.5130873918533325,
      0.5130321979522705,
      0.51297926902771,
      0.512925922870636,
      0.5128723978996277,
      0.5128182172775269,
      0.5127648711204529,
      0.5127115845680237,
      0.5126611590385437,
      0.5126045942306519,
      0.5125513672828674,
      0.5124984383583069,
      0.512444257736206,
      0.5123909711837769,
      0.5123411417007446,
      0.5122861266136169,
      0.5122315287590027,
      0.5121778249740601,
      0.5121244192123413,
      0.5120771527290344,
      0.5120179653167725,
      0.5119661092758179,
      0.5119170546531677,
      0.5118592381477356,
      0.5118052959442139,
      0.5117528438568115,
      0.5116996169090271,
      0.5116491913795471,
      0.5115927457809448,
      0.5115438103675842,
      0.5114859938621521,
      0.511435329914093,
      0.5113798379898071,
      0.5113278031349182,
      0.511275053024292,
      0.511221170425415,
      0.511167585849762,
      0.5111159682273865,
      0.5110617876052856,
      0.5110093355178833,
      0.5109557509422302,
      0.5109058022499084,
      0.5108498334884644]
```

```
weights = model.get_weights()
print(weights[0])
print(weights[1])
```

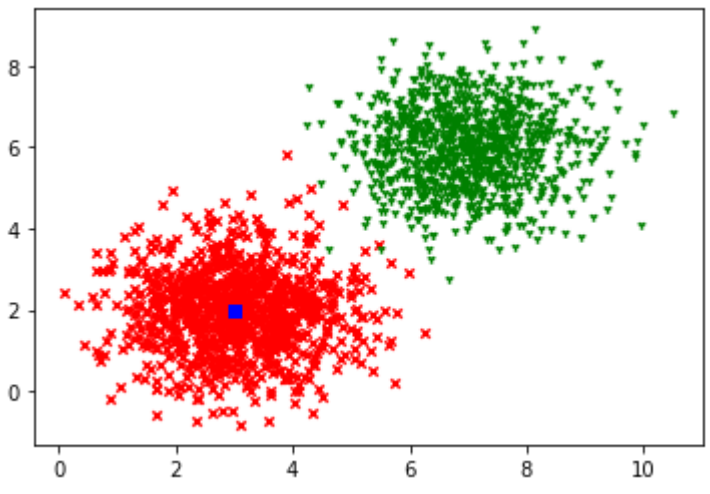
```
     [[-0.11989459]
      [ 0.42002136]]
     [-0.36382034]
```

```
plt.scatter(np.arange(epochs),Loss)
plt.show()
```

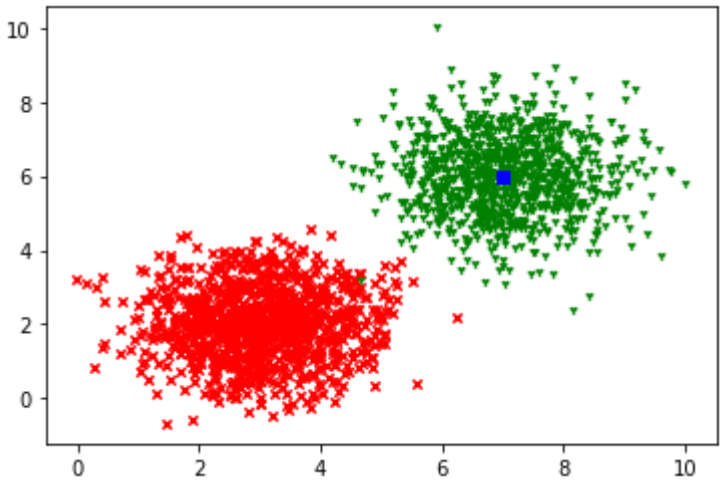



```
x = 3.0
```

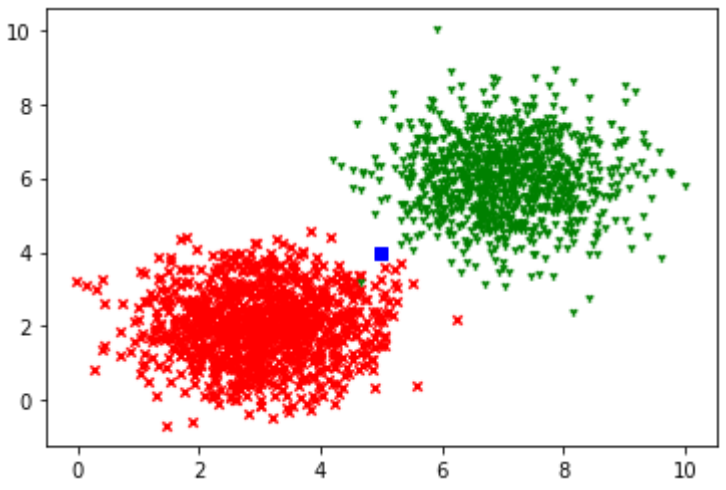
```
y = 2.0
plt.scatter(x_label_1, y_label_1, c='r', marker='x', s=20)
plt.scatter(x_label_2, y_label_2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
```



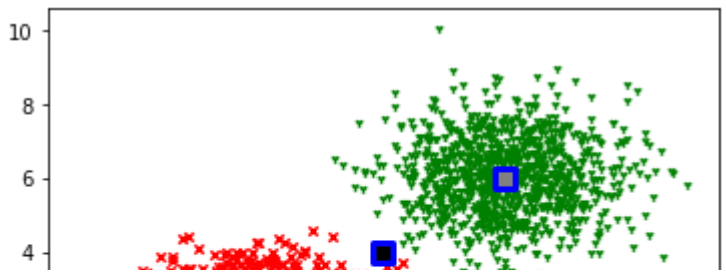

```
x = 7.0
y = 6.0
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
plt.scatter([x],[y],c='b', marker='s')
plt.show()
```




```
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()
```




```
plt.scatter(x_label1, y_label1, c='r', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='g', marker='1', s=20)
x = 3.0
y = 2.0
plt.scatter([x],[y],c = 'w', marker='s', linewidths = 3, edgecolor ="blue", s = 100)
x = 7.0
y = 6.0
plt.scatter([x],[y],c = 'grey', marker='s', linewidths = 3, edgecolor ="blue", s = 100)
x = 5.0
y = 4.0
plt.scatter([x],[y],c = 'black', marker='s', linewidths = 3, edgecolor ="blue", s = 100)
plt.show()
```

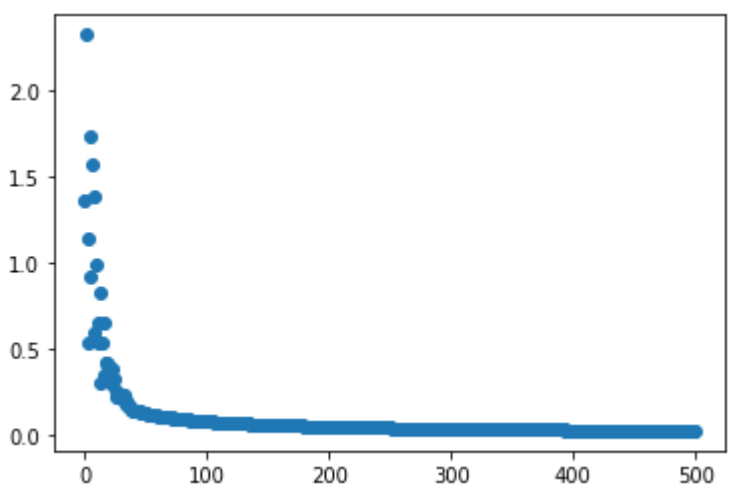



```
print(model.predict([[3, 2]]))
print(model.predict([[7.0, 6.]]))
print(model.predict([[5, 4]]))
```

```
[[0.5291017]]
[[0.7886898]]
[[0.6719002]]
```

```
# v1
Loss = []
epochs = 500
learning_rate = 0.5
print("epochs =", epochs, "and learning rate =", learning_rate)

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

h = model.fit(data_points, labels, verbose = 0, epochs = epochs, batch_size = 1000)
Loss = h.history['loss']

weights = model.get_weights()
print("")
print("weights:")
print(weights[0])
print(weights[1])

print("")
plt.scatter(np.arange(epochs), Loss)
plt.show()
```

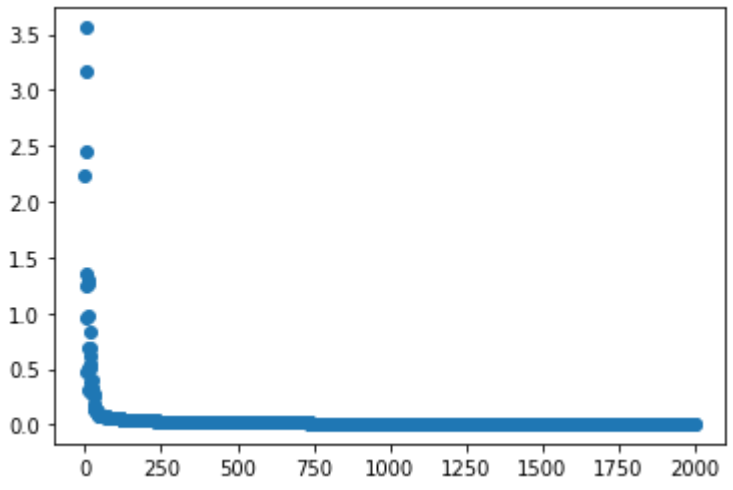
```
epochs = 500 and learning rate = 0.5

weights:
[[1.0692374]
 [1.3554108]]
[-10.540627]
```




```
# v2
Loss = []
epochs = 2000
learning_rate = 0.7
print("epochs =", epochs, "and learning rate =", learning_rate)

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

h = model.fit(data_points, labels, verbose = 0, epochs = epochs, batch_size = 1000)
Loss = h.history['loss']

weights = model.get_weights()
print("")
print("weights:")
print(weights[0])
print(weights[1])

print("")
plt.scatter(np.arange(epochs), Loss)
```

```
plt.show()
```

```
epochs = 2000 and learning rate = 0.7

weights:
[[1.7372426]
 [1.9460917]]
[-16.398281]
```




```
# v3
Loss = []
epochs = 500
learning_rate = 0.001
print("epochs =", epochs, "and learning rate =", learning_rate)

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

h = model.fit(data_points, labels, verbose = 0, epochs = epochs, batch_size = 1000)
Loss = h.history['loss']

weights = model.get_weights()
print("")
print("weights:")
print(weights[0])
print(weights[1])

print("")
plt.scatter(np.arange(epochs), Loss)
plt.show()
```

```
epochs = 500 and learning rate = 0.001

weights:
[[-0.24790922]
 [ 0.520994  ]]
[-0.089828]
```