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

Zbiór danych:

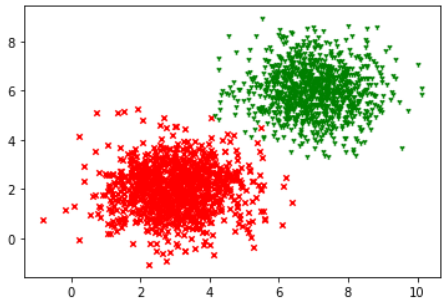
```
[0]*10+[1]*10
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

```
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))
labels
```

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

```
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()
```




```
x_label1
```

```
 2.42027896,  3.09101074,  2.61363546,  2.09213425,  3.13681344,
 3.69794471,  2.73347598,  4.71508056,  2.54209537,  2.64367723,
 2.38117936,  1.80988081,  3.76041107,  2.35846555,  4.52869014,
 3.96083196,  4.21504037,  1.73071861,  1.54467375,  4.04041403,
 4.58336129,  2.54463861,  2.21138333,  3.59740399,  3.23570412,
 2.41160963,  2.53060023,  4.29282929,  3.74001069,  4.29143829,
 2.93270975,  5.28299855,  2.53355768,  0.83565538,  2.39401222,
 3.29257984,  3.02692245,  3.27398502,  3.615339  ,  3.2091535 ,
 0.9047001 ,  1.18422325,  3.4703065 ,  3.68115907,  4.31330536,
 4.12329236,  3.38752732,  2.06913875,  2.08557016,  1.93364259,
 4.10344941,  3.45886475,  4.7767349 ,  3.566521  ,  2.1884025 ,
 4.0707907 ,  1.96158552,  1.44738038,  4.26247498,  3.1668759 ,
 1.42643187,  3.5864296 ,  3.66256136,  1.86540971,  2.7121205 ,
 2.68800272,  3.46826016,  1.7489892 ,  2.67608637,  3.05825031,
 2.2383278 ,  1.21951999,  1.09585847,  1.09857112,  4.04513128,
 2.87306927,  3.1313114 ,  5.07555118,  3.28804273,  3.25548669,
 3.70756971,  1.75960399,  1.37141827,  2.42529627,  3.54644035,
 2.13611275,  2.91727513,  2.12311509,  2.8880284 ,  4.25453206,
 3.10237815,  5.39023   ,  2.57274364,  2.33308433,  1.6419919 ,
 2.8235609 ,  3.28590359,  5.407156  ,  3.35569047,  4.45722253,
 2.88502207,  1.97760597,  2.43845307,  2.98172607,  5.58697182,
 6.08065636,  1.1332135 ,  3.51585287,  4.41911187,  3.10993523,
 2.65019885,  4.96098645,  2.78616511,  1.98038616,  3.89628498,
 2.52236131,  1.50615482,  4.84797792,  3.98770964,  2.37573124,
 4.14028755,  1.3440299 ,  2.49853568,  1.48649723,  2.22602984,
 2.5838681 ,  2.71690621,  1.17372432,  2.55335507,  2.62363487,
 3.23421279,  2.96063358,  1.8682517 ,  3.69844924,  3.70237193,
 3.79110874,  4.62054471,  3.49871944,  1.79623476,  2.90072449,
 1.91975639,  3.89692528,  4.57316504,  2.08373416,  2.39727946,
 2.71983967,  6.18611761,  4.30453198,  2.50760898,  3.63109176,
 1.19658471,  2.52071689,  3.78593019,  3.66157436,  3.57849846,
 3.21530427,  1.96474847,  3.38871853,  3.16194121,  2.72702964,
-0.17045086,  2.0043698 ,  3.90601934,  0.22938353,  2.48864553,
 3.10424151,  2.97185525,  3.43966725,  3.4687294 ,  3.74391323,
 2.96019092,  2.92649012,  2.15893784,  3.07184527,  4.00228948,
 4.45260642,  3.38465494,  3.70086845,  3.08429144,  3.95568098,
 4.02677601,  4.25926368,  4.33025968,  2.93564206,  3.88898888,
 3.71924285,  3.31411241,  2.61650288,  3.91192615,  3.87556946,
 3.01985175,  2.23067481,  3.50484197,  2.81292899,  3.18039627,
 4.45734685,  1.86645973,  2.45394286,  2.21275811,  1.87187686,
 1.52942237,  2.09281005,  3.69840349,  2.93985122,  3.42216206,
 2.68945607,  1.86222682,  1.75724501,  2.38835371,  2.97142638,
 2.16489355,  3.92981562,  2.26687506,  3.78556718,  2.96352252,
 1.81694687,  2.34374356,  3.92752836,  3.81102157,  2.7680302 ,
 1.026952  ,  2.76567575,  1.68441185,  3.86832336,  4.48929247,
 4.3020407 ,  4.07764791,  2.64327082,  4.33568511,  3.30688498,
 2.74579111,  2.73710252,  2.43711626,  2.47355018,  3.07576097,
 5.15578343,  3.18177936,  3.73419705,  3.37416922,  3.94660701,
 2.63969614,  2.45063339,  1.70665735,  3.21541991,  3.06432232,
 2.15339718,  2.6633739 ,  0.23953811,  3.34246951,  4.58010307,
 2.70044254,  4.36589883,  1.52598402,  1.97821326,  2.90427714,
 1.52488726,  3.95410411,  3.05426184,  3.17875069,  1.990036  ,
 1.53990217,  2.8649003 ,  2.3393242 ,  2.85338414,  3.8021988 ,
 1.43171767,  3.587037  ,  3.00033389,  2.57902748,  2.64260444])
```

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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 1)                 3         

=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

Przygotowanie danych:

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

```
array([[1.44941336, 0.46515802],
       [1.35782757, 2.26612285],
       [3.78022886, 2.27151385],
       ...,
       [5.02367742, 4.31663031],
       [7.14511973, 4.72852997],
       [7.58910918, 6.05561945]])
```

Proces **uczenia**:

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

```
50/50 [==============================] - 0s 2ms/step - loss: 0.0366 - val_loss: 0.0430
Epoch 72/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0362 - val_loss: 0.0341
Epoch 73/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0359 - val_loss: 0.0364
Epoch 74/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0355 - val_loss: 0.0346
Epoch 75/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0352 - val_loss: 0.0393
Epoch 76/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0350 - val_loss: 0.0344
Epoch 77/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0346 - val_loss: 0.0360
Epoch 78/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0343 - val_loss: 0.0389
Epoch 79/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0340 - val_loss: 0.0399
Epoch 80/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0338 - val_loss: 0.0365
Epoch 81/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0336 - val_loss: 0.0331
Epoch 82/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0332 - val_loss: 0.0361
Epoch 83/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0329 - val_loss: 0.0394
Epoch 84/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0326 - val_loss: 0.0343
Epoch 85/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0324 - val_loss: 0.0374
Epoch 86/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0320 - val_loss: 0.0297
Epoch 87/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0319 - val_loss: 0.0375
Epoch 88/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0316 - val_loss: 0.0321
Epoch 89/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0314 - val_loss: 0.0346
Epoch 90/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0312 - val_loss: 0.0336
Epoch 91/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0311 - val_loss: 0.0351
Epoch 92/100
50/50 [==============================] - 0s 2ms/step - loss: 0.0307 - val_loss: 0.0370
Epoch 93/100
```

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

```
[0.6807949542999268,
 0.4435085952281952,
 0.3558792173862457,
 0.2999304533004761,
 0.25762686133384705,
 0.22564348578453064,
 0.20140166580677032,
 0.1830454021692276,
 0.16692863404750824,
 0.15461122989654541,
 0.14313337206840515,
 0.13526585698127747,
 0.12587985396385193,
 0.11961708217859268,
 0.11317002028226852,
 0.10783258080482483,
 0.10268449038267136,
```

```
 0.09880365431308746,
 0.09430186450481415,
 0.09058410674333572,
 0.08777061104774475,
 0.08471404016017914,
 0.08183244615793228,
 0.07964075356721878,
 0.07707184553146362,
 0.075049489736557,
 0.07289909571409225,
 0.07067001610994339,
 0.06896388530731201,
 0.06746569275856018,
 0.06565023213624954,
 0.06433440744876862,
 0.06275233626365662,
 0.06114765629172325,
 0.06028449535369873,
 0.058852847665548325,
 0.05778280273079872,
 0.05678950622677803,
 0.05555161461234093,
 0.05458059906959534,
 0.053727105259895325,
 0.052589401602745056,
 0.05181742087006569,
 0.05108642578125,
 0.050221994519233704,
 0.04936468228697777,
 0.048720866441726685,
 0.04794470965862274,
 0.04724324122071266,
 0.046590402722358704,
 0.04597128927707672,
 0.04517819732427597,
 0.04480615630745888,
 0.0440940335392952,
 0.04360105097293854,
 0.04299492388963699,
 0.04270831122994423,
 0.042044606059788661
```

Sprawdźmy jakie są **wartości wag**:

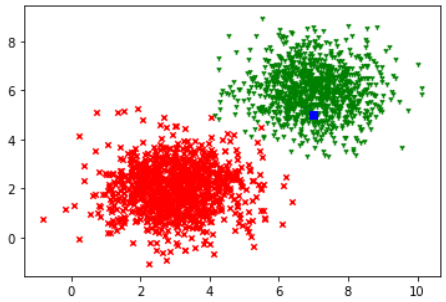
```
weights = model.get_weights()

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

```
[[1.0017886]
 [1.3446786]]
[-10.618045]
```

```
plt.scatter(np.arange(epochs),h.history['loss'])
plt.scatter(np.arange(epochs),h.history['val_loss'],c='r')
plt.show()
```

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

```
x=7.0
y=5.0
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 79ms/step
array([[0.95762384]], dtype=float32)
```

✓ 0 s ukończono o 15:55