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
```

Regresja liniowa

[ ] *Скрыто 2 ячейки.*

Regresja logistyczna

Przykład - mecz szachowy

[ ] *Скрыто 12 ячеек.*

Przykład - 2 gangi

[ ] *Скрыто 14 ячеек.*

Podział na **zbiory treningowy** i **walidacyjny**

[ ] *Скрыто 6 ячеек.*

Regresja **softmax**

[ ] *Скрыто 16 ячеек.*

Zadania

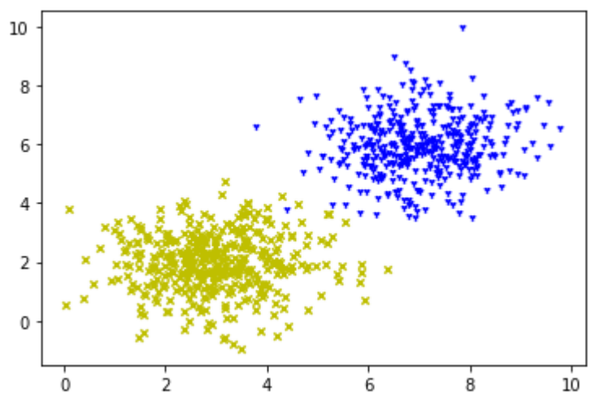
zad.1

```
x_label1 = np.random.normal(3, 1, (400, 1))
y_label1 = np.random.normal(2, 1, (400, 1))
x_label2 = np.random.normal(7, 1, (400, 1))
y_label2 = np.random.normal(6, 1, (400, 1))
```

```
data_label1 = np.concatenate([x_label1, y_label1],axis=1)
data_label2 = np.concatenate([x_label2, y_label2],axis=1)
points = np.concatenate([data_label1, data_label2], axis=0)


# Kodowanie one-hot
labels = np.array([[0.,]] * len(data_label1) + [[1.]] * len(data_label2))

print(points.shape, labels.shape)
```

```
(800, 2) (800, 1)
```

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

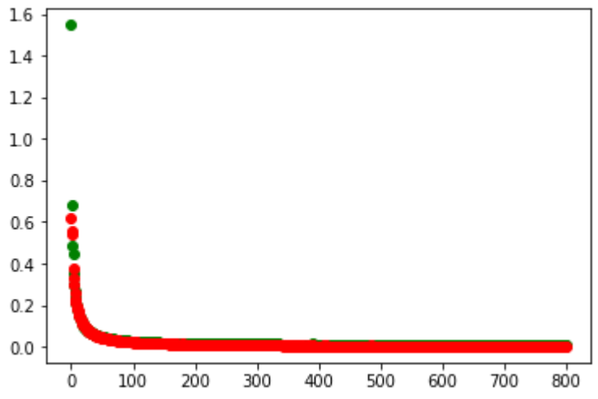



```
arr = np.arange(points.shape[0])
np.random.shuffle(arr)
points = points[arr, :]
labels = labels[arr, :]


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

opt = keras.optimizers.Adam(learning_rate=0.1)
model.compile(loss='binary_crossentropy', optimizer = opt, metrics=['accuracy'])
model.summary()
```

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

```
epochs = 800
h = model.fit(points,labels, verbose=0, validation_split=0.2, epochs=epochs, batch_size=100)


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

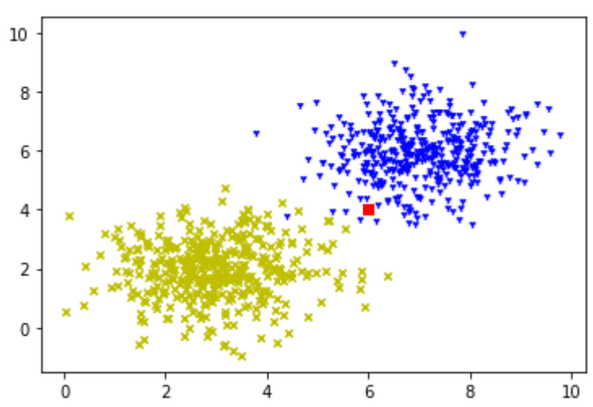
```
plt.scatter(np.arange(epochs), h.history['loss'],c = 'g')
plt.scatter(np.arange(epochs), h.history['val_loss'],c = 'r')
plt.show()
```




```
model.predict([[6, 4]])
```

```
array([[1.]], dtype=float32)
```

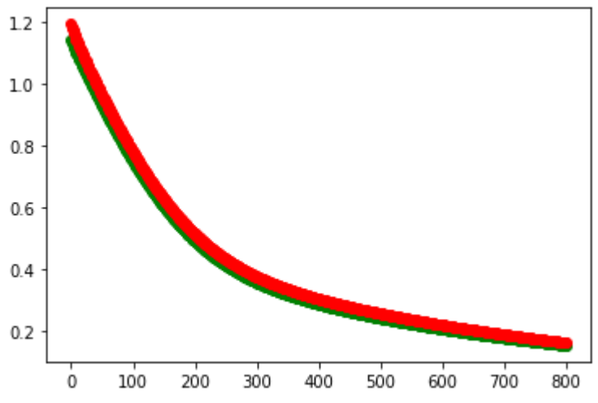
```
x=6.0
y=4.0
plt.scatter(x_label1, y_label1, c='y', marker='x', s=20)
plt.scatter(x_label2, y_label2, c='b', marker='1', s=20)
plt.scatter([x],[y],c='r', marker='s')
plt.show()
```



inne parametry:

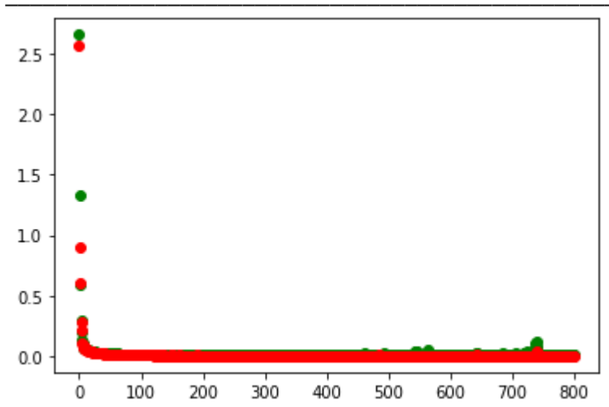
```
# keras Adam 0.001
model = Sequential()
model.add(Dense(units = 1, use_bias = True, input_dim = 2, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.001)
model.compile(loss='binary_crossentropy', optimizer = opt, metrics=['accuracy'])

epochs = 800
h = model.fit(points,labels, verbose=0, validation_split=0.2, epochs=epochs, batch_size=100)

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

```
plt.show()
```



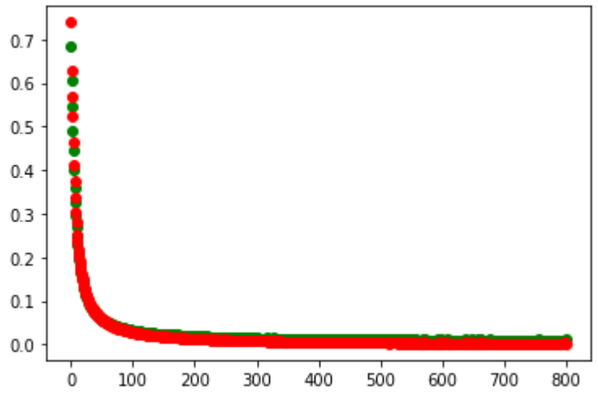

```
# keras Adam 0.5
model = Sequential()
model.add(Dense(units = 1, use_bias = True, input_dim = 2, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.5)
model.compile(loss='binary_crossentropy', optimizer = opt, metrics=['accuracy'])
epochs = 800
h = model.fit(points,labels, verbose=0, validation_split=0.2, epochs=epochs, batch_size=100)

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

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

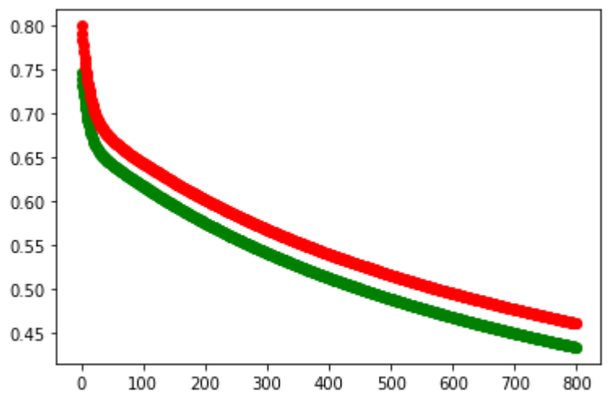



```
# keras Adam 0.05
model = Sequential()
model.add(Dense(units = 1, use_bias = True, input_dim = 2, activation = "softmax"))

opt = keras.optimizers.Adam(learning_rate=0.05)
```

```
model.compile(loss='binary_crossentropy', optimizer = opt, metrics=['accuracy'])
epochs = 800
h = model.fit(points,labels, verbose=0, validation_split=0.2, epochs=epochs, batch_size=100)

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




```
# keras SGD 0.001
model = Sequential()
model.add(Dense(units = 1, use_bias = True, input_dim = 2, activation = "softmax"))

opt = keras.optimizers.SGD(learning_rate=0.001)
model.compile(loss='binary_crossentropy', optimizer = opt, metrics=['accuracy'])
epochs = 800
h = model.fit(points,labels, verbose=0, validation_split=0.2, epochs=epochs, batch_size=100)

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




```
model.predict([[6, 4]])

     array([[1.]], dtype=float32)


# keras SGD 0.5
model = Sequential()
model.add(Dense(units = 1, use_bias = True, input_dim = 2, activation = "softmax"))
```

```
opt = keras.optimizers.SGD(learning_rate=0.5)
model.compile(loss='binary_crossentropy', optimizer = opt, metrics=['accuracy'])

epochs = 800
h = model.fit(points,labels, verbose=0, validation_split=0.2, epochs=epochs, batch_size=100)

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