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

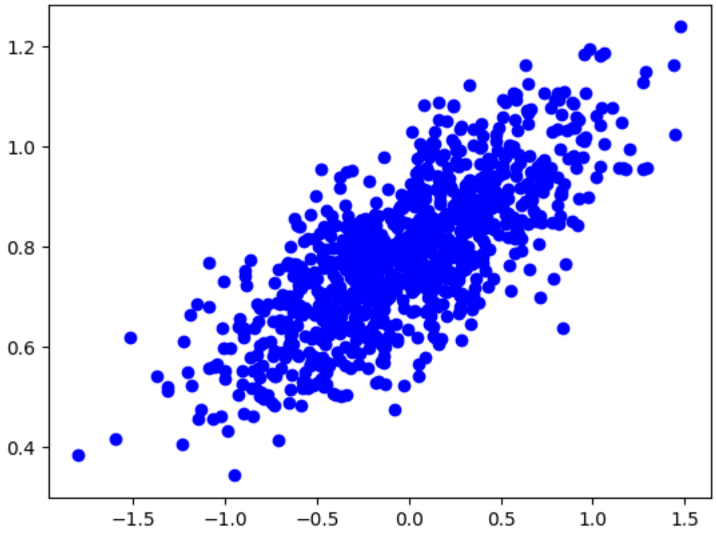
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
```

```
number_of_points = 1000
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='b')
plt.show()
```




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

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

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=1, activation = "linear"))
```

Definiujemy **optymalizator** i **błąd** (średni błąd kwadratowy - MSE). **Współczynnik uczenia = 0.1**

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

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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 2

=================================================================
Total params: 2 (8.00 Byte)
Trainable params: 2 (8.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Proces **uczenia**:

```
epochs = 1000
h = model.fit(real_x,real_y, verbose=0, epochs=epochs, batch_size=100)
```

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

```
 0.009601863101124763,
 0.009614231064915657,
 0.009596983902156353,
 0.009612131863832474,
 0.00959497969597578,
 0.009604258462786674,
 0.009593440219759941,
 0.009594074450433254,
 0.009598399512469769,
 0.009607110172510147,
 0.009609358385205269,
 0.009602859616279602,
 0.009609098546206951,
 0.009608713909983635,
 0.009608577936887741,
 0.009605595842003822,
 0.009591517969965935,
 0.009597528725862503,
 0.009592460468411446,
 0.009615587070584297,
 0.009608644060790539,
 0.009612919762730598,
 0.009593640454113483,
 0.009598343633115292,
 0.00960729643702507,
 0.009608603082597256,
 0.009597192518413067,
 0.009598864242434502,
 0.009595069102942944,
 0.009616900235414505,
 0.009605251252651215,
 0.009605229832231998,
 0.009607142768800259,
 0.009612842462956905,
 0.009605638682842255,
 0.009594103321433067,
 0.00959739275276661,
 0.009600712917745113,
 0.009601416066288948,
 0.009605111554265022,
 0.00959677156060934,
 0.009598590433597565,
 0.00962732918560505,
 0.009596430696547031,
 0.009604223072528839,
 0.009606084786355495,
 0.009592304937541485,
 0.009608170948922634,
 0.00959798227995634,
 0.00960279256105423,
 0.009617241099476814,
 0.009605804458260536,
 0.009610090404748917,
 0.009594223462045193,
 0.009599345736205578,
 0.009620950557291508,
 0.009593555703759193,
 0.009587252512574196]
```

Sprawdźmy jakie są **wartości wag**:

```
weights = model.get_weights()

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

```
0.21978667
0.7881504
```

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