```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
import pandas as pd
%matplotlib inline
from sklearn.model_selection import *
from sklearn.preprocessing import *


data_frame = pd.read_csv('winequality-red.csv', parse_dates=True)


data_frame.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```
data_frame.groupby('quality').count().reset_index()
```

	quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	3	10	10	10	10	10	10	10	10	10	10	10
1	4	53	53	53	53	53	53	53	53	53	53	53
2	5	681	681	681	681	681	681	681	681	681	681	681
3	6	638	638	638	638	638	638	638	638	638	638	638
4	7	199	199	199	199	199	199	199	199	199	199	199
5	8	18	18	18	18	18	18	18	18	18	18	18

```
data_frame['quality'].replace(to_replace={3: 0, 4: 1, 5: 2, 6: 3, 7: 4, 8: 5}, inplace=True)


X = data_frame[['fixed acidity','volatile acidity','citric acid','residual sugar','chlorides',
'free sulfur dioxide','total sulfur dioxide','density','pH','sulphates','alcohol']]
Y = data_frame['quality']


X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=24)
```

```
s = StandardScaler()
X_train = s.fit_transform(X_train)
X_test = s.transform(X_test)


model = tf.keras.models.Sequential([
  keras.layers.Dense(units=128, input_shape=(X_train.shape[1],), activation='relu'),
  keras.layers.Dense(units=64, activation='relu'),
  keras.layers.Dense(units=32, activation='relu'),
  keras.layers.Dense(units=6, activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy', optimizer='sgd',  metrics=['accuracy'])
model.summary()
```

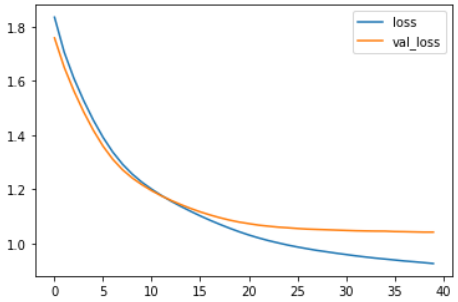
```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_23 (Dense)            (None, 128)               1536

 dense_24 (Dense)            (None, 64)                8256

 dense_25 (Dense)            (None, 32)                2080

 dense_26 (Dense)            (None, 6)                 198

=================================================================
Total params: 12,070
Trainable params: 12,070
Non-trainable params: 0
_________________________________________________________________
```

```
h = model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=40, batch_size=64)
```

```
Epoch 6/40
20/20 [==============================] - 0s 3ms/step - loss: 1.3910 - accuracy: 0.4793 - val_loss: 1.3585 - val_accuracy: 0.4906
Epoch 7/40
20/20 [==============================] - 0s 3ms/step - loss: 1.3369 - accuracy: 0.4957 - val_loss: 1.3104 - val_accuracy: 0.4875
Epoch 8/40
20/20 [==============================] - 0s 3ms/step - loss: 1.2923 - accuracy: 0.5043 - val_loss: 1.2719 - val_accuracy: 0.4938
Epoch 9/40
20/20 [==============================] - 0s 4ms/step - loss: 1.2559 - accuracy: 0.5129 - val_loss: 1.2415 - val_accuracy: 0.5031
Epoch 10/40
20/20 [==============================] - 0s 3ms/step - loss: 1.2258 - accuracy: 0.5293 - val_loss: 1.2166 - val_accuracy: 0.5125
Epoch 11/40
20/20 [==============================] - 0s 4ms/step - loss: 1.2001 - accuracy: 0.5434 - val_loss: 1.1947 - val_accuracy: 0.5219
Epoch 12/40
20/20 [==============================] - 0s 4ms/step - loss: 1.1770 - accuracy: 0.5512 - val_loss: 1.1755 - val_accuracy: 0.5344
Epoch 13/40
20/20 [==============================] - 0s 3ms/step - loss: 1.1560 - accuracy: 0.5575 - val_loss: 1.1588 - val_accuracy: 0.5188
Epoch 14/40
20/20 [==============================] - 0s 4ms/step - loss: 1.1368 - accuracy: 0.5590 - val_loss: 1.1434 - val_accuracy: 0.5375
Epoch 15/40
20/20 [==============================] - 0s 3ms/step - loss: 1.1188 - accuracy: 0.5575 - val_loss: 1.1292 - val_accuracy: 0.5344
```

```
20/20 [==============================] - 0s 3ms/step - loss: 1.0313 - accuracy: 0.5731 - val_loss: 1.0731 - val_accuracy: 0.5469
Epoch 22/40
20/20 [==============================] - 0s 3ms/step - loss: 1.0205 - accuracy: 0.5833 - val_loss: 1.0676 - val_accuracy: 0.5531
Epoch 23/40
20/20 [==============================] - 0s 3ms/step - loss: 1.0107 - accuracy: 0.5825 - val_loss: 1.0638 - val_accuracy: 0.5500
Epoch 24/40
20/20 [==============================] - 0s 3ms/step - loss: 1.0022 - accuracy: 0.5880 - val_loss: 1.0603 - val_accuracy: 0.5656
Epoch 25/40
20/20 [==============================] - 0s 3ms/step - loss: 0.9941 - accuracy: 0.5911 - val_loss: 1.0578 - val_accuracy: 0.5688
Epoch 26/40
20/20 [==============================] - 0s 5ms/step - loss: 0.9871 - accuracy: 0.6005 - val_loss: 1.0552 - val_accuracy: 0.5656
Epoch 27/40
20/20 [==============================] - 0s 3ms/step - loss: 0.9806 - accuracy: 0.6028 - val_loss: 1.0530 - val_accuracy: 0.5625
Epoch 28/40
20/20 [==============================] - 0s 4ms/step - loss: 0.9744 - accuracy: 0.6020 - val_loss: 1.0517 - val_accuracy: 0.5625
Epoch 29/40
20/20 [==============================] - 0s 3ms/step - loss: 0.9693 - accuracy: 0.6036 - val_loss: 1.0505 - val_accuracy: 0.5625
Epoch 30/40
20/20 [==============================] - 0s 3ms/step - loss: 0.9640 - accuracy: 0.6020 - val_loss: 1.0492 - val_accuracy: 0.5625
Epoch 31/40
20/20 [==============================] - 0s 4ms/step - loss: 0.9592 - accuracy: 0.6013 - val_loss: 1.0480 - val_accuracy: 0.5656
Epoch 32/40
20/20 [==============================] - 0s 4ms/step - loss: 0.9544 - accuracy: 0.6059 - val_loss: 1.0468 - val_accuracy: 0.5688
Epoch 33/40
20/20 [==============================] - 0s 4ms/step - loss: 0.9501 - accuracy: 0.6052 - val_loss: 1.0461 - val_accuracy: 0.5688
Epoch 34/40
20/20 [==============================] - 0s 3ms/step - loss: 0.9460 - accuracy: 0.6067 - val_loss: 1.0455 - val_accuracy: 0.5656
```

```
plt.plot(h.history['loss'], label='loss')
plt.plot(h.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
```




```
ModelLoss, ModelAccuracy = model.evaluate(X_test, y_test)

print("Loss")
print(ModelLoss)
print("Accuracy")
print(ModelAccuracy)
```

```
10/10 [==============================] - 0s 2ms/step - loss: 1.0418 - accuracy: 0.5781
Loss
1.04180109500885
Accuracy
0.578125
```

✓ 0 s ukończono o 14:29