```
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=16, activation='relu'),
  keras.layers.Dense(units=6, activation='softmax')
])

model.compile(loss='sparse_categorical_crossentropy', optimizer='sgd',  metrics=['accuracy'])
model.summary()
```

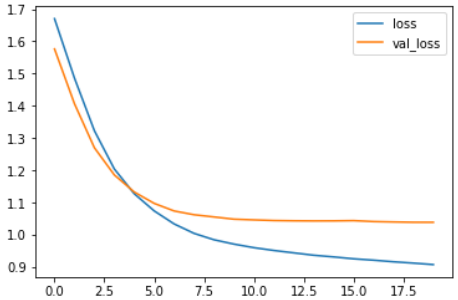
```
Model: "sequential_59"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_296 (Dense)           (None, 128)               1536      

 dense_297 (Dense)           (None, 64)                8256      

 dense_298 (Dense)           (None, 32)                2080      

 dense_299 (Dense)           (None, 16)                528       

 dense_300 (Dense)           (None, 6)                 102       

=================================================================
Total params: 12,502
Trainable params: 12,502
Non-trainable params: 0
_________________________________________________________________
```

```
h = model.fit(X_train, y_train, validation_data=(X_test,y_test), epochs=20)
```

```
Epoch 1/20
40/40 [==============================] - 1s 16ms/step - loss: 1.6704 - accuracy: 0.4636 - val_loss: 1.5764 - val_accuracy: 0.4531
Epoch 2/20
40/40 [==============================] - 0s 3ms/step - loss: 1.4861 - accuracy: 0.4957 - val_loss: 1.4065 - val_accuracy: 0.4969
Epoch 3/20
40/40 [==============================] - 0s 3ms/step - loss: 1.3226 - accuracy: 0.5348 - val_loss: 1.2697 - val_accuracy: 0.5188
Epoch 4/20
40/40 [==============================] - 0s 3ms/step - loss: 1.2038 - accuracy: 0.5598 - val_loss: 1.1853 - val_accuracy: 0.5437
Epoch 5/20
40/40 [==============================] - 0s 3ms/step - loss: 1.1269 - accuracy: 0.5637 - val_loss: 1.1323 - val_accuracy: 0.5594
Epoch 6/20
40/40 [==============================] - 0s 2ms/step - loss: 1.0736 - accuracy: 0.5794 - val_loss: 1.0970 - val_accuracy: 0.5531
Epoch 7/20
40/40 [==============================] - 0s 3ms/step - loss: 1.0332 - accuracy: 0.5887 - val_loss: 1.0734 - val_accuracy: 0.5594
Epoch 8/20
40/40 [==============================] - 0s 3ms/step - loss: 1.0038 - accuracy: 0.5973 - val_loss: 1.0617 - val_accuracy: 0.5719
Epoch 9/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9840 - accuracy: 0.6020 - val_loss: 1.0549 - val_accuracy: 0.5688
Epoch 10/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9705 - accuracy: 0.6067 - val_loss: 1.0479 - val_accuracy: 0.5813
Epoch 11/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9595 - accuracy: 0.6067 - val_loss: 1.0457 - val_accuracy: 0.5813
Epoch 12/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9508 - accuracy: 0.6036 - val_loss: 1.0437 - val_accuracy: 0.5813
Epoch 13/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9434 - accuracy: 0.6067 - val_loss: 1.0429 - val_accuracy: 0.5813
```

```
Epoch 14/20
40/40 [==============================] - 0s 2ms/step - loss: 0.9360 - accuracy: 0.6091 - val_loss: 1.0425 - val_accuracy: 0.5844
Epoch 15/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9308 - accuracy: 0.6122 - val_loss: 1.0427 - val_accuracy: 0.5781
Epoch 16/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9249 - accuracy: 0.6091 - val_loss: 1.0434 - val_accuracy: 0.6062
Epoch 17/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9206 - accuracy: 0.6145 - val_loss: 1.0406 - val_accuracy: 0.5813
Epoch 18/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9157 - accuracy: 0.6130 - val_loss: 1.0393 - val_accuracy: 0.5938
Epoch 19/20
40/40 [==============================] - 0s 2ms/step - loss: 0.9115 - accuracy: 0.6185 - val_loss: 1.0383 - val_accuracy: 0.5969
Epoch 20/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9069 - accuracy: 0.6200 - val_loss: 1.0383 - val_accuracy: 0.6000
```

```
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: 1.0383 - accuracy: 0.6000
Loss
1.038275957107544
Accuracy
0.6000000238418579
```

✓ 0 s ukończono o 15:28