```
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()
```

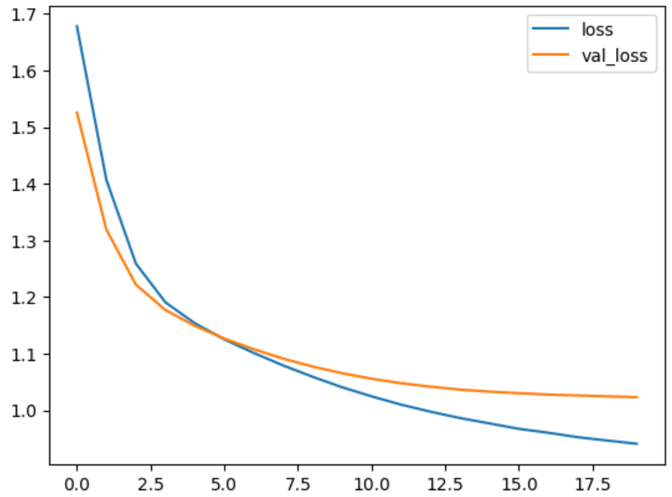
```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_45 (Dense)            (None, 128)               1536      
                                                                 
 dense_46 (Dense)            (None, 64)                8256      
                                                                 
 dense_47 (Dense)            (None, 32)                2080      
                                                                 
 dense_48 (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=20, batch_size=32)
```

```
Epoch 1/20
40/40 [==============================] - 1s 7ms/step - loss: 1.6777 - accuracy: 0.3972 - val_loss: 1.5256 - val_accuracy: 0.4625
Epoch 2/20
40/40 [==============================] - 0s 3ms/step - loss: 1.4068 - accuracy: 0.5027 - val_loss: 1.3190 - val_accuracy: 0.4938
Epoch 3/20
40/40 [==============================] - 0s 3ms/step - loss: 1.2593 - accuracy: 0.5168 - val_loss: 1.2221 - val_accuracy: 0.4969
Epoch 4/20
40/40 [==============================] - 0s 3ms/step - loss: 1.1910 - accuracy: 0.5457 - val_loss: 1.1773 - val_accuracy: 0.5250
Epoch 5/20
40/40 [==============================] - 0s 3ms/step - loss: 1.1539 - accuracy: 0.5590 - val_loss: 1.1494 - val_accuracy: 0.5375
Epoch 6/20
40/40 [==============================] - 0s 3ms/step - loss: 1.1255 - accuracy: 0.5614 - val_loss: 1.1268 - val_accuracy: 0.5562
Epoch 7/20
40/40 [==============================] - 0s 3ms/step - loss: 1.1015 - accuracy: 0.5700 - val_loss: 1.1081 - val_accuracy: 0.5562
Epoch 8/20
40/40 [==============================] - 0s 3ms/step - loss: 1.0793 - accuracy: 0.5747 - val_loss: 1.0914 - val_accuracy: 0.5562
Epoch 9/20
40/40 [==============================] - 0s 3ms/step - loss: 1.0594 - accuracy: 0.5770 - val_loss: 1.0774 - val_accuracy: 0.5562
Epoch 10/20
40/40 [==============================] - 0s 3ms/step - loss: 1.0409 - accuracy: 0.5817 - val_loss: 1.0657 - val_accuracy: 0.5625
Epoch 11/20
40/40 [==============================] - 0s 3ms/step - loss: 1.0248 - accuracy: 0.5825 - val_loss: 1.0559 - val_accuracy: 0.5625
Epoch 12/20
40/40 [==============================] - 0s 3ms/step - loss: 1.0102 - accuracy: 0.5848 - val_loss: 1.0480 - val_accuracy: 0.5781
Epoch 13/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9978 - accuracy: 0.5919 - val_loss: 1.0419 - val_accuracy: 0.5813
Epoch 14/20
40/40 [==============================] - 0s 4ms/step - loss: 0.9866 - accuracy: 0.5989 - val_loss: 1.0367 - val_accuracy: 0.5844
Epoch 15/20
40/40 [==============================] - 0s 4ms/step - loss: 0.9771 - accuracy: 0.6091 - val_loss: 1.0331 - val_accuracy: 0.5844
Epoch 16/20
40/40 [==============================] - 0s 3ms/step - loss: 0.9675 - accuracy: 0.6106 - val_loss: 1.0304 - val_accuracy: 0.5844
Epoch 17/20
40/40 [==============================] - 0s 5ms/step - loss: 0.9606 - accuracy: 0.6138 - val_loss: 1.0279 - val_accuracy: 0.5813
Epoch 18/20
40/40 [==============================] - 0s 4ms/step - loss: 0.9527 - accuracy: 0.6177 - val_loss: 1.0264 - val_accuracy: 0.5844
Epoch 19/20
```

```
40/40 [==============================] - 0s 6ms/step - loss: 0.9468 - accuracy: 0.6255 - val_loss: 1.0248 - val_accuracy: 0.5875
Epoch 20/20
40/40 [==============================] - 0s 4ms/step - loss: 0.9412 - accuracy: 0.6224 - val_loss: 1.0233 - val_accuracy: 0.5844
```

```
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 3ms/step - loss: 1.0233 - accuracy: 0.5844
Loss
1.0233267545700073
Accuracy
0.5843750238418579
```