```
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.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
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

```
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()
```

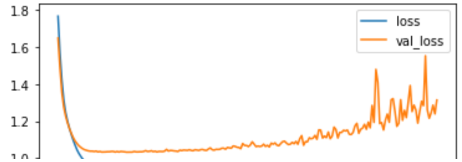
```
Model: "sequential_76"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_381 (Dense)           (None, 128)               1536

 dense_382 (Dense)           (None, 64)                8256

 dense_383 (Dense)           (None, 32)                2080

 dense_384 (Dense)           (None, 16)                528

 dense_385 (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=200,batch_size=32)
```

```
Epoch 1/200
40/40 [==============================] - 1s 7ms/step - loss: 1.7651 - accuracy: 0.2901 - val_loss: 1.6471 - val_accuracy: 0.4313
Epoch 2/200
40/40 [==============================] - 0s 3ms/step - loss: 1.5696 - accuracy: 0.4629 - val_loss: 1.4876 - val_accuracy: 0.4594
Epoch 3/200
```

```
40/40 [==============================] - 0s 3ms/step - loss: 1.4227 - accuracy: 0.4566 - val_loss: 1.3566 - val_accuracy: 0.4375
Epoch 4/200
40/40 [==============================] - 0s 3ms/step - loss: 1.3152 - accuracy: 0.4543 - val_loss: 1.2765 - val_accuracy: 0.4531
Epoch 5/200
40/40 [==============================] - 0s 3ms/step - loss: 1.2470 - accuracy: 0.4652 - val_loss: 1.2270 - val_accuracy: 0.4750
Epoch 6/200
40/40 [==============================] - 0s 3ms/step - loss: 1.2001 - accuracy: 0.4973 - val_loss: 1.1907 - val_accuracy: 0.4875
Epoch 7/200
40/40 [==============================] - 0s 3ms/step - loss: 1.1612 - accuracy: 0.5496 - val_loss: 1.1601 - val_accuracy: 0.5063
Epoch 8/200
40/40 [==============================] - 0s 2ms/step - loss: 1.1276 - accuracy: 0.5582 - val_loss: 1.1341 - val_accuracy: 0.5031
Epoch 9/200
40/40 [==============================] - 0s 3ms/step - loss: 1.0971 - accuracy: 0.5661 - val_loss: 1.1113 - val_accuracy: 0.5156
Epoch 10/200
40/40 [==============================] - 0s 3ms/step - loss: 1.0698 - accuracy: 0.5825 - val_loss: 1.0928 - val_accuracy: 0.5344
Epoch 11/200
40/40 [==============================] - 0s 3ms/step - loss: 1.0451 - accuracy: 0.5958 - val_loss: 1.0793 - val_accuracy: 0.5437
Epoch 12/200
40/40 [==============================] - 0s 3ms/step - loss: 1.0254 - accuracy: 0.6013 - val_loss: 1.0672 - val_accuracy: 0.5625
Epoch 13/200
40/40 [==============================] - 0s 3ms/step - loss: 1.0097 - accuracy: 0.6036 - val_loss: 1.0586 - val_accuracy: 0.5688
Epoch 14/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9953 - accuracy: 0.6052 - val_loss: 1.0512 - val_accuracy: 0.5656
Epoch 15/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9838 - accuracy: 0.6013 - val_loss: 1.0461 - val_accuracy: 0.5656
Epoch 16/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9740 - accuracy: 0.6005 - val_loss: 1.0426 - val_accuracy: 0.5719
Epoch 17/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9649 - accuracy: 0.6052 - val_loss: 1.0399 - val_accuracy: 0.5688
Epoch 18/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9563 - accuracy: 0.6028 - val_loss: 1.0385 - val_accuracy: 0.5781
Epoch 19/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9490 - accuracy: 0.6099 - val_loss: 1.0385 - val_accuracy: 0.5938
Epoch 20/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9433 - accuracy: 0.6145 - val_loss: 1.0366 - val_accuracy: 0.5938
Epoch 21/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9346 - accuracy: 0.6114 - val_loss: 1.0388 - val_accuracy: 0.5781
Epoch 22/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9303 - accuracy: 0.6185 - val_loss: 1.0347 - val_accuracy: 0.5813
Epoch 23/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9247 - accuracy: 0.6091 - val_loss: 1.0369 - val_accuracy: 0.5969
Epoch 24/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9207 - accuracy: 0.6122 - val_loss: 1.0338 - val_accuracy: 0.5938
Epoch 25/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9154 - accuracy: 0.6145 - val_loss: 1.0341 - val_accuracy: 0.5969
Epoch 26/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9105 - accuracy: 0.6138 - val_loss: 1.0345 - val_accuracy: 0.6031
Epoch 27/200
40/40 [==============================] - 0s 3ms/step - loss: 0.9061 - accuracy: 0.6192 - val_loss: 1.0355 - val_accuracy: 0.6000
Epoch 28/200
40/40 [==============================] - 0s 2ms/step - loss: 0.9021 - accuracy: 0.6200 - val_loss: 1.0354 - val_accuracy: 0.6031
Epoch 29/200
40/40 [==============================] - 0s 2ms/step - loss: 0.8975 - accuracy: 0.6255 - val_loss: 1.0370 - val_accuracy: 0.5875
```

```
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: 1.3137 - accuracy: 0.5906
Loss
1.3136932849884033
Accuracy
0.590624988079071
```