```
import numpy as np

# makes printing more human-friendly
np.set_printoptions(precision=3, suppress=True)
```

```
colab=True
if colab:
    from google.colab import drive
    drive.mount('/content/drive')
    with open('/content/drive/MyDrive/Colab Notebooks/data/winequality-white.csv', 'r') as f:
        data = np.genfromtxt(f, delimiter=',', skip_header=1)
else:
    with open('winequality-white.csv', 'r') as f:
        data = np.genfromtxt(f, delimiter=',', skip_header=1)

X = data[:, :-1]
y = data[:, -1].astype(int)

print('num_samples, num_features', X.shape)
print('unique labels', np.unique(y))
```

```
     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
     num_samples, num_features (4898, 11)
     unique labels [3 4 5 6 7 8 9]
```

```
mean = np.mean(X, axis=0)
std = np.std(X, axis=0)
normalized_features = (X - mean) / std
```

```
labels_one_hot = np.eye(7)[y - 3]
```

```
from sklearn.model_selection import train_test_split

X_temp, X_test, y_temp, y_test = train_test_split(normalized_features, labels_one_hot, test_size=0.2, stratify=y)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.25, stratify=y_temp.argmax(axis=1))
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Define the model
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),  # Input layer
    Dense(64, activation='relu'),                                    # Hidden layer
    Dense(7, activation='softmax')                                   # Output layer
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

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

      dense_1 (Dense)             (None, 64)                8256

      dense_2 (Dense)             (None, 7)                 455

     =================================================================
     Total params: 10247 (40.03 KB)
     Trainable params: 10247 (40.03 KB)
     Non-trainable params: 0 (0.00 Byte)
     _________________________________________________________________
```

```
history = model.fit(X_train, y_train,
                    epochs=50,
                    batch_size=32,
                    validation_data=(X_val, y_val))
```

```
Epoch 22/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8610 - accuracy: 0.6389 - val_loss: 1.0487 - val_accuracy: 0.5398
Epoch 23/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8574 - accuracy: 0.6389 - val_loss: 1.0521 - val_accuracy: 0.5408
Epoch 24/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8419 - accuracy: 0.6555 - val_loss: 1.0493 - val_accuracy: 0.5429
Epoch 25/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8360 - accuracy: 0.6535 - val_loss: 1.0480 - val_accuracy: 0.5418
Epoch 26/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8285 - accuracy: 0.6552 - val_loss: 1.0569 - val_accuracy: 0.5500
Epoch 27/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8171 - accuracy: 0.6576 - val_loss: 1.0589 - val_accuracy: 0.5357
Epoch 28/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8132 - accuracy: 0.6593 - val_loss: 1.0779 - val_accuracy: 0.5082
Epoch 29/50
92/92 [==============================] - 0s 3ms/step - loss: 0.8016 - accuracy: 0.6719 - val_loss: 1.0714 - val_accuracy: 0.5418
Epoch 30/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7987 - accuracy: 0.6644 - val_loss: 1.0793 - val_accuracy: 0.5316
Epoch 31/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7839 - accuracy: 0.6675 - val_loss: 1.0668 - val_accuracy: 0.5449
Epoch 32/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7767 - accuracy: 0.6814 - val_loss: 1.0720 - val_accuracy: 0.5449
Epoch 33/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7677 - accuracy: 0.6852 - val_loss: 1.0728 - val_accuracy: 0.5520
Epoch 34/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7608 - accuracy: 0.6790 - val_loss: 1.0857 - val_accuracy: 0.5490
Epoch 35/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7489 - accuracy: 0.6933 - val_loss: 1.0966 - val_accuracy: 0.5388
Epoch 36/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7450 - accuracy: 0.6940 - val_loss: 1.0943 - val_accuracy: 0.5378
Epoch 37/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7367 - accuracy: 0.7032 - val_loss: 1.0905 - val_accuracy: 0.5439
Epoch 38/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7281 - accuracy: 0.6957 - val_loss: 1.0908 - val_accuracy: 0.5500
Epoch 39/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7245 - accuracy: 0.7032 - val_loss: 1.0873 - val_accuracy: 0.5480
Epoch 40/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7146 - accuracy: 0.7148 - val_loss: 1.0830 - val_accuracy: 0.5469
Epoch 41/50
92/92 [==============================] - 0s 3ms/step - loss: 0.7066 - accuracy: 0.7114 - val_loss: 1.0909 - val_accuracy: 0.5408
Epoch 42/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.7195 - val_loss: 1.1015 - val_accuracy: 0.5449
Epoch 43/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6888 - accuracy: 0.7144 - val_loss: 1.1292 - val_accuracy: 0.5449
Epoch 44/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6823 - accuracy: 0.7246 - val_loss: 1.1203 - val_accuracy: 0.5490
Epoch 45/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6736 - accuracy: 0.7243 - val_loss: 1.1032 - val_accuracy: 0.5510
Epoch 46/50
92/92 [==============================] - 0s 5ms/step - loss: 0.6654 - accuracy: 0.7270 - val_loss: 1.1197 - val_accuracy: 0.5337
Epoch 47/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6558 - accuracy: 0.7369 - val_loss: 1.1303 - val_accuracy: 0.5367
Epoch 48/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6565 - accuracy: 0.7328 - val_loss: 1.1331 - val_accuracy: 0.5439
Epoch 49/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6487 - accuracy: 0.7403 - val_loss: 1.1508 - val_accuracy: 0.5541
Epoch 50/50
92/92 [==============================] - 0s 4ms/step - loss: 0.6370 - accuracy: 0.7447 - val_loss: 1.1377 - val_accuracy: 0.5296
```

```
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test accuracy: {test_accuracy*100:.2f}%")
```

```
31/31 [==============================] - 0s 2ms/step - loss: 1.0602 - accuracy: 0.5796
Test accuracy: 57.96%
```