```
from google.colab import drive
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping

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)

X = data[:, :-1]
y = data[:, -1]
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
y_one_hot = to_categorical(y - 3)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


X_train, X_temp, y_train, y_temp = train_test_split(X_scaled, y_one_hot, test_size=0.4, stratify=y_one_hot.argmax(1))
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, stratify=y_temp.argmax(1))


# Build the model
H = 64  # Number of neurons in the hidden layers, replace with desired number
model = Sequential([
    Dense(H, activation='relu', input_shape=(11,)),
    Dense(H, activation='relu'),
    Dense(H, activation='relu'),
    Dense(H, activation='relu'),
    Dense(7, activation='softmax')
])

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


#train the model

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,
    verbose=1
)

test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)

final_loss_value = history.history['loss'][-1]
final_val_loss_value = history.history['val_loss'][-1]

print(f"Final training loss value: {final_loss_value}")
print(f"Final validation loss value: {final_val_loss_value}")
print(f"Test accuracy: {test_accuracy}")
```

```
Epoch 83/100
92/92 [==============================] - 0s 3ms/step - loss: 0.1348 - accuracy: 0.9585 - val_loss: 2.5393 - val_accuracy: 0.5612
Epoch 84/100
92/92 [==============================] - 0s 5ms/step - loss: 0.1119 - accuracy: 0.9643 - val_loss: 2.6641 - val_accuracy: 0.5531
Epoch 85/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1310 - accuracy: 0.9561 - val_loss: 2.6650 - val_accuracy: 0.5551
Epoch 86/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1471 - accuracy: 0.9493 - val_loss: 2.6604 - val_accuracy: 0.5551
Epoch 87/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1551 - accuracy: 0.9415 - val_loss: 2.6178 - val_accuracy: 0.5714
Epoch 88/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1219 - accuracy: 0.9643 - val_loss: 2.6064 - val_accuracy: 0.5602
Epoch 89/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1083 - accuracy: 0.9673 - val_loss: 2.7490 - val_accuracy: 0.5714
Epoch 90/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1338 - accuracy: 0.9537 - val_loss: 2.6535 - val_accuracy: 0.5694
Epoch 91/100
92/92 [==============================] - 0s 4ms/step - loss: 0.0913 - accuracy: 0.9755 - val_loss: 2.7366 - val_accuracy: 0.5684
Epoch 92/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1027 - accuracy: 0.9694 - val_loss: 2.8171 - val_accuracy: 0.5735
Epoch 93/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1090 - accuracy: 0.9660 - val_loss: 2.8312 - val_accuracy: 0.5378
Epoch 94/100
92/92 [==============================] - 0s 4ms/step - loss: 0.1082 - accuracy: 0.9622 - val_loss: 2.8127 - val_accuracy: 0.5612
Epoch 95/100
92/92 [==============================] - 0s 4ms/step - loss: 0.0942 - accuracy: 0.9711 - val_loss: 2.8625 - val_accuracy: 0.5755
Epoch 96/100
92/92 [==============================] - 0s 3ms/step - loss: 0.0942 - accuracy: 0.9721 - val_loss: 2.9467 - val_accuracy: 0.5888
Epoch 97/100
92/92 [==============================] - 0s 3ms/step - loss: 0.0966 - accuracy: 0.9704 - val_loss: 2.9282 - val_accuracy: 0.5755
Epoch 98/100
92/92 [==============================] - 0s 3ms/step - loss: 0.0908 - accuracy: 0.9731 - val_loss: 2.9677 - val_accuracy: 0.5602
Epoch 99/100
92/92 [==============================] - 0s 3ms/step - loss: 0.1197 - accuracy: 0.9595 - val_loss: 3.0244 - val_accuracy: 0.5663
Epoch 100/100
92/92 [==============================] - 0s 3ms/step - loss: 0.1074 - accuracy: 0.9700 - val_loss: 3.1013 - val_accuracy: 0.5469
Final training loss value: 0.10744686424732208
Final validation loss value: 3.101270914077759
Test accuracy: 0.5642856955528259
```

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Flatten, Reshape
from tensorflow.keras.models import Model

fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

x_train = x_train.reshape((-1, 784))
x_test = x_test.reshape((-1, 784))

input_img = Input(shape=(784,))
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)

decoded = Dense(128, activation='relu')(encoded)
decoded = Dense(784, activation='sigmoid')(decoded)

autoencoder = Model(input_img, decoded)

autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

# Train the model
autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))

encoder = Model(input_img, encoded)

encoded_imgs = encoder.predict(x_test)
```

```
235/235 [==============================] - 4s 18ms/step - loss: 0.2700 - val_loss: 0.2722
Epoch 26/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2697 - val_loss: 0.2722
Epoch 27/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2696 - val_loss: 0.2719
Epoch 28/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2693 - val_loss: 0.2716
Epoch 29/50
235/235 [==============================] - 5s 20ms/step - loss: 0.2691 - val_loss: 0.2714
Epoch 30/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2689 - val_loss: 0.2713
Epoch 31/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2687 - val_loss: 0.2713
Epoch 32/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2686 - val_loss: 0.2709
Epoch 33/50
235/235 [==============================] - 5s 19ms/step - loss: 0.2684 - val_loss: 0.2708
Epoch 34/50
235/235 [==============================] - 3s 14ms/step - loss: 0.2683 - val_loss: 0.2706
Epoch 35/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2682 - val_loss: 0.2705
Epoch 36/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2680 - val_loss: 0.2704
Epoch 37/50
235/235 [==============================] - 4s 17ms/step - loss: 0.2679 - val_loss: 0.2703
Epoch 38/50
235/235 [==============================] - 4s 15ms/step - loss: 0.2677 - val_loss: 0.2702
Epoch 39/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2677 - val_loss: 0.2702
Epoch 40/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2676 - val_loss: 0.2704
Epoch 41/50
235/235 [==============================] - 3s 15ms/step - loss: 0.2674 - val_loss: 0.2698
Epoch 42/50
235/235 [==============================] - 4s 18ms/step - loss: 0.2673 - val_loss: 0.2699
Epoch 43/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2673 - val_loss: 0.2697
Epoch 44/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2671 - val_loss: 0.2696
Epoch 45/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2670 - val_loss: 0.2695
Epoch 46/50
235/235 [==============================] - 5s 20ms/step - loss: 0.2669 - val_loss: 0.2695
Epoch 47/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2670 - val_loss: 0.2694
Epoch 48/50
235/235 [==============================] - 3s 12ms/step - loss: 0.2668 - val_loss: 0.2692
Epoch 49/50
235/235 [==============================] - 3s 13ms/step - loss: 0.2667 - val_loss: 0.2692
Epoch 50/50
235/235 [==============================] - 5s 19ms/step - loss: 0.2666 - val_loss: 0.2691
313/313 [==============================] - 1s 2ms/step
```