```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models, preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

```
df = pd.read_csv("./ecommerceDataset.csv", header=None)
df.columns = ['label', 'text']
df.head()
```

	label	text
0	Household	Paper Plane Design Framed Wall Hanging Motivat...
1	Household	SAF 'Floral' Framed Painting (Wood, 30 inch x ...
2	Household	SAF 'UV Textured Modern Art Print Framed' Pain...
3	Household	SAF Flower Print Framed Painting (Synthetic, 1...
4	Household	Incredible Gifts India Wooden Happy Birthday U...

```
df.dropna(inplace=True)
df.drop_duplicates(inplace=True)
```

```
text = df.text
label = df.label

tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(text)

vocab_size = len(tokenizer.word_index) + 1
vocab_size
print("the vocab size is {}".format(vocab_size))
```

```
the vocab size is 92268
```

```
max_length = 400

token_to_seq = tokenizer.texts_to_sequences(text)

padded_text = preprocessing.sequence.pad_sequences(token_to_seq,
                                                        truncating='post',
                                                        padding='post',
                                                        maxlen=max_length)
```

```
label = LabelEncoder().fit_transform(label)
```

```
array([3, 3, 3, 3, 3])
```

```
X_train, X_test, y_train, y_test = train_test_split(padded_text, label,
                                                    test_size=0.2,
                                                    random_state=42,
                                                    shuffle=True,
                                                    stratify=label)
y_train_enc = keras.utils.to_categorical(y_train)
y_test_enc = keras.utils.to_categorical(y_test)
```

```
model = models.Sequential()
model.add(layer=layers.Embedding(input_dim=vocab_size, output_dim=128, input_length=max_length, mask_zero=True))
model.add(layer=layers.GRU(units=64, activation=tf.nn.relu))
model.add(layer=layers.BatchNormalization())
model.add(layer=layers.Dense(units=128, activation=tf.nn.relu))
model.add(layer=layers.Dense(units=128, activation=tf.nn.relu))
model.add(layer=layers.Dropout(0.2))
model.add(layer=layers.Dense(units=4, activation=tf.nn.softmax))

optimizer = keras.optimizers.Adam(learning_rate=0.001)
loss = keras.losses.CategoricalCrossentropy()
```

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

early_stopping = keras.callbacks.EarlyStopping(patience=10)

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 400, 128)          11810304

 gru_1 (GRU)                 (None, 64)                37248

 batch_normalization_1 (Batc  (None, 64)               256
 hNormalization)

 dense_3 (Dense)             (None, 128)               8320

 dense_4 (Dense)             (None, 128)               16512

 dropout_1 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 4)                 516

=================================================================
Total params: 11,873,156
Trainable params: 11,873,028
Non-trainable params: 128
_________________________________________________________________
```

```
history = model.fit(X_train, y_train_enc,
                    epochs=5,
                    batch_size=128,
                    validation_split=0.2,
                    callbacks=[early_stopping])
```

```
Epoch 1/5
139/139 [==============================] - 106s 745ms/step - loss: 0.7843 - accuracy: 0.6844 - val_loss: 0.9837 - val_accura
Epoch 2/5
139/139 [==============================] - 97s 695ms/step - loss: 0.2002 - accuracy: 0.9444 - val_loss: 0.5434 - val_accurac
Epoch 3/5
139/139 [==============================] - 97s 700ms/step - loss: 0.0861 - accuracy: 0.9759 - val_loss: 0.2959 - val_accurac
Epoch 4/5
139/139 [==============================] - 97s 700ms/step - loss: 0.0454 - accuracy: 0.9884 - val_loss: 0.2756 - val_accurac
Epoch 5/5
139/139 [==============================] - 99s 714ms/step - loss: 0.0223 - accuracy: 0.9944 - val_loss: 0.3079 - val_accurac
```

```
from sklearn.metrics import *

predictions = np.argmax(model.predict(X_test), axis=1)
print(classification_report(predictions, y_test))
```

```
174/174 [==============================] - 9s 50ms/step
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      1228
           1       0.95      0.98      0.96      1095
           2       0.88      0.90      0.89      1034
           3       0.95      0.91      0.93      2204

    accuracy                           0.92      5561
   macro avg       0.92      0.93      0.92      5561
weighted avg       0.92      0.92      0.92      5561
```

✓ 9s completed at 12:51 PM ● ×