```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras import layers, models

from sklearn.preprocessing import LabelEncoder
import numpy as np
import pandas as pd

np.random.seed(1234)

df = pd.read_csv('./ecommerceDataset.csv', header=None)
df.dropna(inplace=True)
df.drop_duplicates(inplace=True)
df
```

	0	1
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...
...	...	...
50402	Electronics	Micromax Bharat 5 Plus Zero impact on visual d...
50403	Electronics	Microsoft Lumia 550 8GB 4G Black Microsoft lum...
50407	Electronics	Microsoft Lumia 535 (Black, 8GB) Colour:Black ...
50408	Electronics	Karbonn Titanium Wind W4 (White) Karbonn Titan...
50410	Electronics	Nokia Lumia 530 (Dual SIM, Grey) Colour:Grey ...

27802 rows × 2 columns

```
i = np.random.rand(len(df)) < 0.8
train = df[i]
test = df[~i]
print("train data size: ", train.shape)
print("test data size: ", test.shape)
```

```
     train data size:  (22250, 2)
     test data size:  (5552, 2)
```

```
num_labels = 2
vocab_size = 25000
batch_size = 100

train[1] = train[1].astype(str)

tokenizer = Tokenizer(num_words=vocab_size)
tokenizer.fit_on_texts(train[1])

x_train = tokenizer.texts_to_matrix(train[1], mode='tfidf')
x_test = tokenizer.texts_to_matrix(test[1], mode='tfidf')

encoder = LabelEncoder()
encoder.fit(train[0])
y_train = encoder.transform(train[0])
y_test = encoder.transform(test[0])

print("train shapes:", x_train.shape, y_train.shape)
print("test shapes:", x_test.shape, y_test.shape)
```

```
     <ipython-input-4-e1144b85d17d>:5: SettingWithCopyWarning:
     A value is trying to be set on a copy of a slice from a DataFrame.
     Try using .loc[row_indexer,col_indexer] = value instead

     See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
       train[1] = train[1].astype(str)
     train shapes: (22250, 25000) (22250,)
     test shapes: (5552, 25000) (5552,)
```

```
model = models.Sequential()
model.add(layers.Dense(32, input_dim=vocab_size, kernel_initializer='normal', activation='relu'))
model.add(layers.Dense(1, kernel_initializer='normal', activation='sigmoid'))

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

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=5,
                    verbose=1,
                    validation_split=0.2)
```

```
Epoch 1/5
178/178 [==============================] - 6s 28ms/step - loss: -67.0464 - accuracy: 0.3539 - val_loss: -208.0863 - val_accu
Epoch 2/5
178/178 [==============================] - 4s 24ms/step - loss: -549.1127 - accuracy: 0.4033 - val_loss: -922.7766 - val_acc
Epoch 3/5
178/178 [==============================] - 6s 31ms/step - loss: -1563.9841 - accuracy: 0.4073 - val_loss: -2127.5315 - val_a
Epoch 4/5
178/178 [==============================] - 4s 25ms/step - loss: -3076.4949 - accuracy: 0.4094 - val_loss: -3775.1699 - val_a
Epoch 5/5
178/178 [==============================] - 4s 25ms/step - loss: -5046.2173 - accuracy: 0.4066 - val_loss: -5846.3560 - val_a
```

```
score = model.evaluate(x_test, y_test, batch_size=batch_size, verbose=1)
print('Accuracy: ', score[1])
```

```
56/56 [==============================] - 0s 8ms/step - loss: -6022.2451 - accuracy: 0.3276
Accuracy:  0.3276296854019165
```