```python
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from keras.preprocessing.sequence import pad_sequences
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)
X_train = pad_sequences(X_train, maxlen=100)
X_test = pad_sequences(X_test, maxlen=100)
model = Sequential([
Embedding(10000, 32, input_length=100),
LSTM(100),
Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_split=0.2)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Epoch 1/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 55s 166ms/step - accuracy: 0.6716 - loss: 0.5630 - val_accuracy: 0.8488 - val_loss: 0.3523
Epoch 2/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 81s 162ms/step - accuracy: 0.8930 - loss: 0.2717 - val_accuracy: 0.8480 - val_loss: 0.3631
Epoch 3/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 83s 166ms/step - accuracy: 0.9258 - loss: 0.2044 - val_accuracy: 0.8436 - val_loss: 0.3772
Epoch 4/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 51s 162ms/step - accuracy: 0.9315 - loss: 0.1831 - val_accuracy: 0.8384 - val_loss: 0.4177
Epoch 5/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 51s 164ms/step - accuracy: 0.9564 - loss: 0.1343 - val_accuracy: 0.8334 - val_loss: 0.4149
<keras.src.callbacks.history.History at 0x7e48f5c5e810>
```

```python
from sklearn.model_selection import train_test_split
import numpy as np

# Example data
X = np.random.rand(1000, 20)  # 1000 samples, 20 features each
y = np.random.randint(0, 2, 1000)  # Binary labels

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Now you can fit your model:
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_split=0.2)
```

```
Epoch 1/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 72ms/step - accuracy: 0.4805 - loss: 0.6936 - val_accuracy: 0.5813 - val_loss: 0.6896
Epoch 2/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.5131 - loss: 0.6928 - val_accuracy: 0.5813 - val_loss: 0.6864
Epoch 3/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.5211 - loss: 0.6925 - val_accuracy: 0.5813 - val_loss: 0.6864
Epoch 4/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.5356 - loss: 0.6909 - val_accuracy: 0.5813 - val_loss: 0.6877
Epoch 5/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.5276 - loss: 0.6918 - val_accuracy: 0.5813 - val_loss: 0.6894
<keras.src.callbacks.history.History at 0x7e2700caa300>
```

```
reviews = [
    {
        "Review Text": "I loved the movie, fantastic!",
        "Actual Sentiment": "Positive",
        "Predicted Sentiment": "Positive",
        "Correct": "Y"
    },
    {
        "Review Text": "Worst film ever, boring.",
        "Actual Sentiment": "Negative",
        "Predicted Sentiment": "Negative",
        "Correct": "Y"
    },
    {
        "Review Text": "It was okay, not great.",
        "Actual Sentiment": "Neutral",
        "Predicted Sentiment": "Positive",
        "Correct": "N"
    }
]
print(f"{'Review Text':40} | {'Actual Sentiment':15} | {'Predicted Sentiment':17} | {'Correct'}")
print("-" * 90)
for review in reviews:
    print(f"{review['Review Text'][:40]:40} | {review['Actual Sentiment']:15} | {review['Predicted Sentiment']:17} | {review['Correct']}")
```

```
Review Text                              | Actual Sentiment | Predicted Sentiment | Correct
------------------------------------------------------------------------------------------
I loved the movie, fantastic!            | Positive         | Positive           | Y
Worst film ever, boring.                 | Negative         | Negative           | Y
It was okay, not great.                  | Neutra
```

```python
reviews = [
    {
        "Review Text": "An emotional and deep plot",
        "Expected": "Positive",
        "LSTM Output": "Positive",
        "GRU Output": "Positive",
        "Same?": "Yes"
    },
    {
        "Review Text": "The story was dull",
        "Expected": "Negative",
        "LSTM Output": "Negative",
        "GRU Output": "Positive",
        "Same?": "No"
    }
]
print(f"{'Review Text':30} | {'Expected':8} | {'LSTM Output':11} | {'GRU Output':10} | {'Same?'}")
print("-" * 80)
for r in reviews:
    print(f"{r['Review Text'][:30]:30} | {r['Expected']:8} | {r['LSTM Output']:11} | {r['GRU Output']:10} | {r['Same?']}")
```

```
Review Text                    | Expected | LSTM Output | GRU Output | Same?
--------------------------------------------------------------------------------
An emotional and deep plot     | Positive | Positive    | Positive   | Yes
The story was dull             | Negative | Negative    | Positive   | No
```