3. Use an LSTM-based RNN to perform sentiment analysis on movie reviews using the IMDB dataset. Handle sequence padding, embedding, and explore the effect of GRU vs LSTM.

Sentiment Analysis on IMDB Reviews using LSTM & GRU

□ Objective

Build an RNN model to classify movie reviews as **Positive (1)** or **Negative (0)** using:

- Tokenization
- Sequence Padding
- Word Embedding
- LSTM and GRU comparison

□ Dataset

We use the **IMDB Movie Review Dataset (50,000 reviews)** built directly into Keras.

Step 1: Import Libraries

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, GRU, Dense, Dropout
```

Step 2: Load Dataset

```
vocab_size = 10000   # keep top 10k frequent words
max_len = 200

(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=vocab_size)
```

Step 3: Sequence Padding

```
X_train = pad_sequences(X_train, maxlen=max_len, padding='post')
X_test = pad_sequences(X_test, maxlen=max_len, padding='post')
```

□ This makes all reviews **same length (200 words)**.

Step 4: LSTM Model

```
model_lstm = Sequential([
    Embedding(vocab_size, 128, input_length=max_len),
    LSTM(128, return_sequences=False),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model_lstm.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model_lstm.summary()
```

Step 5: Train LSTM

```
model_lstm.fit(X_train, y_train, epochs=5, batch_size=64,
validation_split=0.2)
```

Step 6: Evaluate LSTM

```
loss_lstm, acc_lstm = model_lstm.evaluate(X_test, y_test)
print("LSTM Accuracy:", acc_lstm)
```

Step 7: GRU Model

```
model_gru = Sequential([
    Embedding(vocab_size, 128, input_length=max_len),
    GRU(128),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model_gru.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model_gru.fit(X_train, y_train, epochs=5, batch_size=64,
validation_split=0.2)
```

Step 8: Evaluate GRU

```
loss_gru, acc_gru = model_gru.evaluate(X_test, y_test)
print("GRU Accuracy:", acc_gru)
```

□ Expected Results

Model	Test Accuracy	Training Speed
LSTM	~87–89%	Slower
GRU	~86–88%	Faster

□ Key Observations

Feature	LSTM	GRU
Gates	3	2
Memory	Long-term stronger	Simpler
Speed	Slower	Faster
Accuracy	Slightly higher	Almost same

□ Conclusion

- **LSTM** handles long dependencies better
- **GRU** is computationally cheaper and faster
- Both perform excellently for IMDB sentiment analysis