Build a Deep learning model using LSTM layer in Keras for IMDB dataset

```
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, SimpleRNN, Dense

# Parameters
max_features = 10000  # Number of words to consider as features
maxlen = 500  # Cuts off texts after this many words (among the max_features most common words)
batch_size = 32
embedding_dim = 50
epochs = 10

# Load and preprocess the data
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)

# Define the model
model = Sequential()
model.add(Embedding(max_features, embedding_dim, input_length=maxlen))
model.add(SimpleRNN(32))  # You can adjust the number of units as needed
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print(f'Test accuracy: {accuracy * 100:.2f}%')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 0s 0us/step
Epoch 1/10
625/625 [==============================] - 81s 127ms/step - loss: 0.5078 - accuracy: 0.7459 - val_loss: 0.4155 - val_accuracy: 0.8208
Epoch 2/10
625/625 [==============================] - 75s 121ms/step - loss: 0.3432 - accuracy: 0.8601 - val_loss: 0.3803 - val_accuracy: 0.8292
Epoch 3/10
625/625 [==============================] - 77s 123ms/step - loss: 0.2860 - accuracy: 0.8866 - val_loss: 0.3630 - val_accuracy: 0.8390
Epoch 4/10
625/625 [==============================] - 75s 121ms/step - loss: 0.2437 - accuracy: 0.9075 - val_loss: 0.3611 - val_accuracy: 0.8494
Epoch 5/10
625/625 [==============================] - 77s 123ms/step - loss: 0.1968 - accuracy: 0.9276 - val_loss: 0.3829 - val_accuracy: 0.8644
Epoch 6/10
625/625 [==============================] - 76s 121ms/step - loss: 0.1735 - accuracy: 0.9369 - val_loss: 0.4426 - val_accuracy: 0.8542
Epoch 7/10
625/625 [==============================] - 77s 124ms/step - loss: 0.1286 - accuracy: 0.9536 - val_loss: 0.4678 - val_accuracy: 0.8350
Epoch 8/10
625/625 [==============================] - 78s 124ms/step - loss: 0.0877 - accuracy: 0.9710 - val_loss: 0.5599 - val_accuracy: 0.8130
Epoch 9/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0668 - accuracy: 0.9777 - val_loss: 0.7149 - val_accuracy: 0.8068
Epoch 10/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0528 - accuracy: 0.9837 - val_loss: 0.6564 - val_accuracy: 0.8100
782/782 [==============================] - 21s 27ms/step - loss: 0.6836 - accuracy: 0.8019
Test accuracy: 80.19%
```

Double-click (or enter) to edit