Pratiksha Madane BE-A Roll No 43

AIM :Binary classification using Deep Neural Networks Example: Classify movie reviews into positive" reviews and "negative" reviews, just based on the text content of the reviews. Use IMDB dataset

In [1]:

```
import numpy as np
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.regularizers import l2
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
```

In [2]:

```
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=10000)
# Preprocess data
x_train = [[word_index if word_index < 10000 else 0 for word_index in sequence] for sequence in x_train]
x_test = [[word_index if word_index < 10000 else 0 for word_index in sequence] for sequence in x_test]
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 34s 2us/step
```

In [3]:

```
x_train = pad_sequences(x_train, maxlen=100)
x_test = pad_sequences(x_test, maxlen=100)
# Define model
model = Sequential()
model.add(Embedding(input_dim=10000, output_dim=128, input_length=100))
model.add(LSTM(128, kernel_regularizer=l2(0.001)))  # Removed one LSTM layer
model.add(Dropout(0.4))  # Increased dropout
model.add(Dense(1, activation='sigmoid'))
```

```
C:\Users\Pratiksha\anaconda3\Lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(
```

In [4]:

```
model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.0003), metrics=['accuracy'])
# Early stopping to prevent overfitting
early_stop = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)
# Train model
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_split=0.2, callbacks=[early_stop])
```

```
Epoch 1/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 118s 352ms/step - accuracy: 0.6284 - loss: 0.7388 - val_accuracy: 0.8256 - val_loss: 0.4333
Epoch 2/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 106s 339ms/step - accuracy: 0.8735 - loss: 0.3369 - val_accuracy: 0.8456 - val_loss: 0.3692
Epoch 3/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 107s 341ms/step - accuracy: 0.9164 - loss: 0.2534 - val_accuracy: 0.8426 - val_loss: 0.3776
Epoch 4/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 142s 342ms/step - accuracy: 0.9295 - loss: 0.2191 - val_accuracy: 0.8428 - val_loss: 0.4060
```

In [5]:

```
loss, acc = model.evaluate(x_test, y_test, batch_size=64)
print(f'Test accuracy: {acc:.4f}, Test loss: {loss:.4f}')
```

```
391/391 ━━━━━━━━━━━━━━━━━━━━ 53s 136ms/step - accuracy: 0.8461 - loss: 0.3737
Test accuracy: 0.8464, Test loss: 0.3730
```

In [ ]: