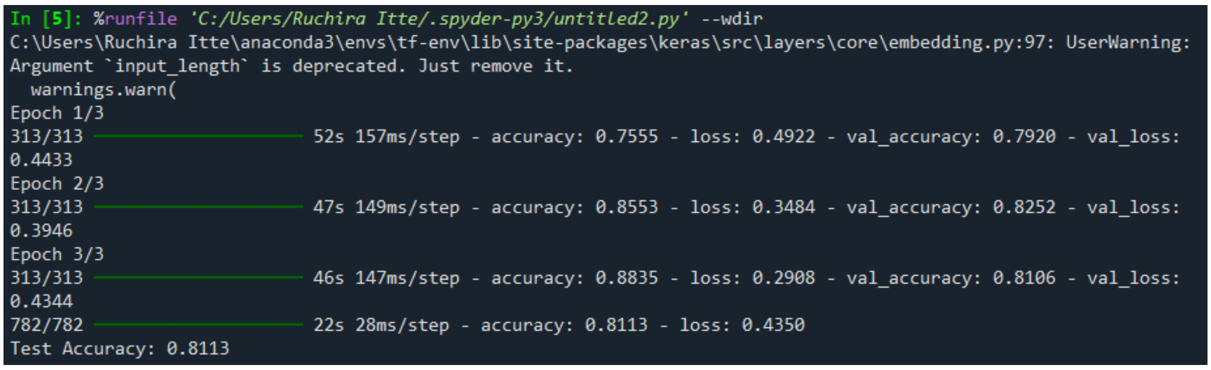
Practical-2

2.A) Aim: Build a NLP model for sentiment analysis

Code:

```
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, Dense

# Load IMDB dataset
num_words = 10000
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=num_words)

# Pad sequences
max_len = 200
x_train = pad_sequences(x_train, maxlen=max_len)
x_test = pad_sequences(x_test, maxlen=max_len)

# Build Model
model = Sequential([
    Embedding(input_dim=num_words, output_dim=128, input_length=max_len),
    LSTM(64, dropout=0.2, recurrent_dropout=0.2),
    Dense(1, activation='sigmoid')
])

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

# Train
model.fit(x_train, y_train, epochs=3, batch_size=64, validation_split=0.2)

# Evaluate
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

# Reverse word index (optional for decoding reviews)
word_index = imdb.get_word_index()
index_offset = 3
word_index = {word: (index + index_offset) for word, index in word_index.items()}
word_index["<PAD>"] = 0
word_index["<START>"] = 1
word_index["<UNK>"] = 2
reverse_word_index = {index: word for word, index in word_index.items()}
```

Output:



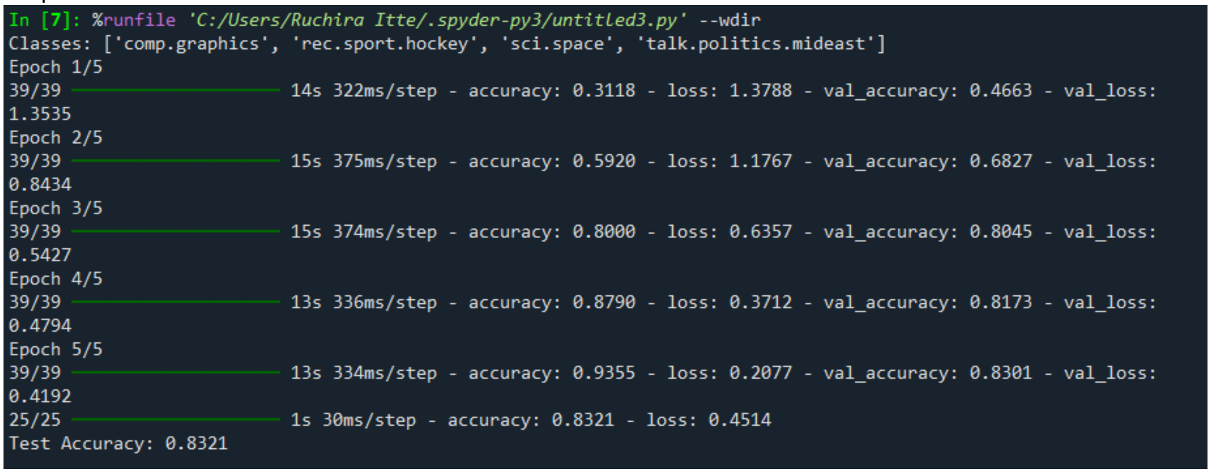
```
In [5]: %runfile 'C:/Users/Ruchira Itte/.spyder-py3/untitled2.py' --wdir
C:\Users\Ruchira Itte\anaconda3\envs\tf-env\lib\site-packages\keras\src\layers\core\embedding.py:97: UserWarning:
Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Epoch 1/3
313/313 ━━━━━━━━━━━━━━━━━━━━ 52s 157ms/step - accuracy: 0.7555 - loss: 0.4922 - val_accuracy: 0.7920 - val_loss:
0.4433
Epoch 2/3
313/313 ━━━━━━━━━━━━━━━━━━━━ 47s 149ms/step - accuracy: 0.8553 - loss: 0.3484 - val_accuracy: 0.8252 - val_loss:
0.3946
Epoch 3/3
313/313 ━━━━━━━━━━━━━━━━━━━━ 46s 147ms/step - accuracy: 0.8835 - loss: 0.2908 - val_accuracy: 0.8106 - val_loss:
0.4344
782/782 ━━━━━━━━━━━━━━━━━━━━ 22s 28ms/step - accuracy: 0.8113 - loss: 0.4350
Test Accuracy: 0.8113
```

2 B) Aim: Build a NLP model for text classification.

Code:

```
import re
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.utils.class_weight import compute_class_weight
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Input
from tensorflow.keras.callbacks import EarlyStopping

# Load subset of Newsgroups
categories = ["sci.space", "comp.graphics", "rec.sport.hockey", "talk.politics.mideast"]
newsgroups = fetch_20newsgroups(
    subset="all",
    categories=categories,
    remove=("headers", "footers", "quotes")
)
texts, labels = newsgroups.data, newsgroups.target
class_names = newsgroups.target_names
print("Classes:", class_names)

# Preprocess text
def clean_text(text):
    text = text.lower()
    text = re.sub(r"[^a-z\s]", " ", text)  # keep only letters
    text = re.sub(r"\s+", " ", text)       # remove extra spaces
    return text.strip()

texts = [clean_text(t) for t in texts]

# Tokenization and Padding
max_words = 10000
max_len = 200
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

X = pad_sequences(sequences, maxlen=max_len)

# Encode labels
encoder = LabelEncoder()
y = encoder.fit_transform(labels)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build LSTM model
model = Sequential([
    Input(shape=(max_len,)),  # explicitly declare input shape
    Embedding(input_dim=max_words, output_dim=128),
    LSTM(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(64, activation="relu"),
    Dropout(0.5),
    Dense(len(class_names), activation="softmax")
])

# Compile
model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

# Train
early_stop = EarlyStopping(monitor="val_loss", patience=2, restore_best_weights=True)
model.fit(X_train, y_train, epochs=5, batch_size=64, validation_split=0.2, callbacks=[early_stop])

# Evaluate
loss, acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {acc:.4f}")
```

output:



```
In [7]: %runfile 'C:/Users/Ruchira Itte/.spyder-py3/untitled3.py' --wdir
Classes: ['comp.graphics', 'rec.sport.hockey', 'sci.space', 'talk.politics.mideast']
Epoch 1/5
39/39 ━━━━━━━━━━━━━━━━━━━━ 14s 322ms/step - accuracy: 0.3118 - loss: 1.3788 - val_accuracy: 0.4663 - val_loss:
1.3535
Epoch 2/5
39/39 ━━━━━━━━━━━━━━━━━━━━ 15s 375ms/step - accuracy: 0.5920 - loss: 1.1767 - val_accuracy: 0.6827 - val_loss:
0.8434
Epoch 3/5
39/39 ━━━━━━━━━━━━━━━━━━━━ 15s 374ms/step - accuracy: 0.8000 - loss: 0.6357 - val_accuracy: 0.8045 - val_loss:
0.5427
Epoch 4/5
39/39 ━━━━━━━━━━━━━━━━━━━━ 13s 336ms/step - accuracy: 0.8790 - loss: 0.3712 - val_accuracy: 0.8173 - val_loss:
0.4794
Epoch 5/5
39/39 ━━━━━━━━━━━━━━━━━━━━ 13s 334ms/step - accuracy: 0.9355 - loss: 0.2077 - val_accuracy: 0.8301 - val_loss:
0.4192
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.8321 - loss: 0.4514
Test Accuracy: 0.8321
```