EX.NO.: 18
DATE: 17.03.2025

SENTIMENT ANALYSIS OF TEXT REVIEWS USING LSTM

To build an LSTM-based sentiment analysis model to classify customer reviews as positive or negative.

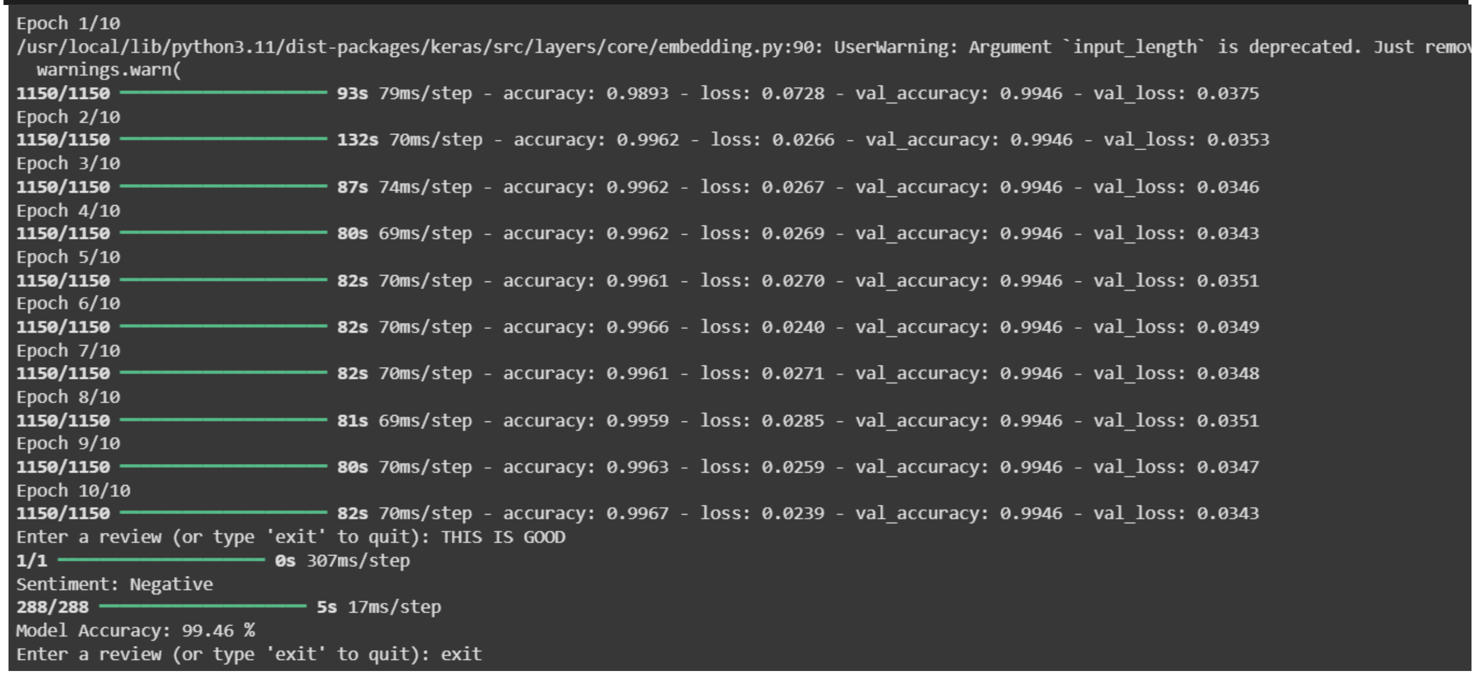
PROCEDURE:

1. Import libraries
2. Dataset loading and handle any hidden spaces
3. Perform text preprocessing
4. Perform tokenization and padding
5. Split data into training and testing
6. Build LSTM model
7. Perform model compilation and training
8. Prompt user input and predict review

CODE AND OUTPUT

```
import re
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
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

# Load the dataset
df = pd.read_csv('IMDB.csv', encoding='utf-8', quoting=3, on_bad_lines='skip')

# Strip any hidden spaces in columns
df.columns = df.columns.str.strip()

# Text cleaning function
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters and numbers
    stop_words =
set("i,me,my,myself,we,our,ours,ourselves,you,your,yours,yourself,yourselves".split(","
))
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)

# Clean the reviews
df['cleaned_review'] = df['review'].fillna('').apply(clean_text)

# Convert 'sentiment' column to binary (0 for negative, 1 for positive)
df['sentiment'] = df['sentiment'].fillna('').apply(lambda x: 1 if str(x).lower() ==
'positive' else 0)
```

```python
# Tokenization
max_words = 5000  # Vocabulary size
max_sequence_length = 100  # Max length of each review
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(df['cleaned_review'])
sequences = tokenizer.texts_to_sequences(df['cleaned_review'])

# Padding sequences
X = pad_sequences(sequences, maxlen=max_sequence_length, padding='post')

# Target variable
y = df['sentiment'].values

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

# LSTM Model
model = Sequential([
    Embedding(input_dim=max_words, output_dim=16, input_length=max_sequence_length),
    LSTM(64, return_sequences=True),
    LSTM(32),
    Dense(1, activation='sigmoid')
])

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

# Train the model with class weights
class_weights = {0: 1.5, 1: 1.0}  # Handle class imbalance
model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1,
validation_data=(X_test, y_test), class_weight=class_weights)

# User input for prediction
while True:
    user_input = input("Enter a review (or type 'exit' to quit): ")
    if user_input.lower() == 'exit':
        break
    user_input_clean = clean_text(user_input)
    user_sequence = tokenizer.texts_to_sequences([user_input_clean])
    user_padded = pad_sequences(user_sequence, maxlen=max_sequence_length,
padding='post')
    prediction = model.predict(user_padded)
    sentiment = "Positive" if prediction > 0.5 else "Negative"
    print("Sentiment:", sentiment)

    # Check accuracy on the test set
    y_pred = (model.predict(X_test) > 0.5).astype(int)
    accuracy = accuracy_score(y_test, y_pred)
    print("Model Accuracy:", round(accuracy * 100, 2), "%")
```



```
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remov
  warnings.warn(
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 93s 79ms/step - accuracy: 0.9893 - loss: 0.0728 - val_accuracy: 0.9946 - val_loss: 0.0375
Epoch 2/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 132s 70ms/step - accuracy: 0.9962 - loss: 0.0266 - val_accuracy: 0.9946 - val_loss: 0.0353
Epoch 3/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 87s 74ms/step - accuracy: 0.9962 - loss: 0.0267 - val_accuracy: 0.9946 - val_loss: 0.0346
Epoch 4/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 80s 69ms/step - accuracy: 0.9962 - loss: 0.0269 - val_accuracy: 0.9946 - val_loss: 0.0343
Epoch 5/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 82s 70ms/step - accuracy: 0.9961 - loss: 0.0270 - val_accuracy: 0.9946 - val_loss: 0.0351
Epoch 6/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 82s 70ms/step - accuracy: 0.9966 - loss: 0.0240 - val_accuracy: 0.9946 - val_loss: 0.0349
Epoch 7/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 82s 70ms/step - accuracy: 0.9961 - loss: 0.0271 - val_accuracy: 0.9946 - val_loss: 0.0348
Epoch 8/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 81s 69ms/step - accuracy: 0.9959 - loss: 0.0285 - val_accuracy: 0.9946 - val_loss: 0.0351
Epoch 9/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 80s 70ms/step - accuracy: 0.9963 - loss: 0.0259 - val_accuracy: 0.9946 - val_loss: 0.0347
Epoch 10/10
1150/1150 ━━━━━━━━━━━━━━━━━━━━ 82s 70ms/step - accuracy: 0.9967 - loss: 0.0239 - val_accuracy: 0.9946 - val_loss: 0.0343
Enter a review (or type 'exit' to quit): THIS IS GOOD
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 307ms/step
Sentiment: Negative
288/288 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step
Model Accuracy: 99.46 %
Enter a review (or type 'exit' to quit): exit
```