```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.14)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.6.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.4)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.7)
```

Importing the dependencies

```
import os
import json

from zipfile import ZipFile
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Embedding,LSTM
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Data Collection - Kaggle API

```
kaggle_dictionary=json.load(open("kaggle.json"))
```

```
kaggle_dictionary.keys()
```

```
dict_keys(['username', 'key'])
```

```
#setup kaggle credentials as environment variables
os.environ["KAGGLE_USERNAME"]= kaggle_dictionary["username"]
os.environ["KAGGLE_KEY"]=kaggle_dictionary["key"]
```

```
!kaggle datasets download -d lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
```

```
Dataset URL: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
License(s): other
Downloading imdb-dataset-of-50k-movie-reviews.zip to /content
 93% 24.0M/25.7M [00:02<00:00, 20.3MB/s]
100% 25.7M/25.7M [00:02<00:00, 11.7MB/s]
```

```
!ls
```

```
imdb-dataset-of-50k-movie-reviews.zip  kaggle.json  sample_data
```

```
# unzip the dataset file
```

```
with ZipFile("imdb-dataset-of-50k-movie-reviews.zip","r") as zip_ref:
  zip_ref.extractall()
```

```
!ls
```

```
'IMDB Dataset.csv'   imdb-dataset-of-50k-movie-reviews.zip   kaggle.json   sample_data
```

Loading the Dataset

```
data=pd.read_csv("/content/IMDB Dataset.csv")
```

```
data.shape
```

```
(50000, 2)
```

```
data.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

```
data.tail()
```

	review	sentiment
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

```
data["sentiment"].value_counts()
```

```
sentiment
positive    25000
negative    25000
Name: count, dtype: int64
```

```
#Encoding the Sentiment column because neural network model won't take labels
data.replace({"sentiment":{"positive":1,"negative":0}}, inplace= True)
```

```
data.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. <br /><br />The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

```
#Splitting data into test data and train data
train_data, test_data=train_test_split(data,test_size=0.2,random_state=42)
```

```
print(train_data.shape)
print(test_data.shape)
```

```
(40000, 2)
(10000, 2)
```

Data Preprocessing

```
#Tokenizing the data - converting review column for model in understandable format

# Tokenize text data
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(train_data["review"])
X_train = pad_sequences(tokenizer.texts_to_sequences(train_data["review"]), maxlen=200)
X_test = pad_sequences(tokenizer.texts_to_sequences(test_data["review"]), maxlen=200)
```

```
print(X_train)
```

```
[[1935    1 1200 ...  205  351 3856]
 [   3 1651  595 ...   89  103    9]
 [   0    0    0 ...    2  710   62]
 ...
 [   0    0    0 ... 1641    2  603]
 [   0    0    0 ...  245  103  125]
 [   0    0    0 ...   70   73 2062]]
```

```
print(X_test)
```

```
[[   0    0    0 ...  995  719  155]
 [  12  162   59 ...  380    7    7]
 [   0    0    0 ...   50 1088   96]
 ...
 [   0    0    0 ...  125  200 3241]
 [   0    0    0 ... 1066    1 2305]
 [   0    0    0 ...    1  332   27]]
```

```
Y_train=train_data["sentiment"]
Y_test=test_data["sentiment"]
```

```
print(Y_train)
```

```
39087    0
30893    0
45278    1
16398    0
13653    0
        ..
11284    1
44732    1
38158    0
860      1
15795    1
Name: sentiment, Length: 40000, dtype: int64
```

```
print(Y_test)
```

```
33553    1
9427     1
199      0
12447    1
39489    0
        ..
28567    0
25079    1
18707    1
15200    0
5857     1
Name: sentiment, Length: 10000, dtype: int64
```

Building LSTM Model

```
#Building LSTM based Neural Network
```

```
model = Sequential()
model.add(Embedding(input_dim=5000,output_dim=128,input_length=200))
model.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
model.add(Dense(1,activation="sigmoid"))
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fa
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 128)          640000

 lstm (LSTM)                 (None, 128)               131584

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 771713 (2.94 MB)
Trainable params: 771713 (2.94 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

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

Training the Model

```
model.fit(X_train,Y_train,epochs=5,batch_size=64,validation_split=0.2)
```

```
Epoch 1/5
500/500 [==============================] - 242s 472ms/step - loss: 0.3910 - accuracy: 0.8241 - val_loss: 0.3144 - val_accuracy: 0.86
Epoch 2/5
500/500 [==============================] - 217s 434ms/step - loss: 0.3170 - accuracy: 0.8673 - val_loss: 0.3396 - val_accuracy: 0.86
Epoch 3/5
500/500 [==============================] - 224s 448ms/step - loss: 0.2333 - accuracy: 0.9080 - val_loss: 0.2926 - val_accuracy: 0.87
Epoch 4/5
500/500 [==============================] - 208s 417ms/step - loss: 0.2055 - accuracy: 0.9203 - val_loss: 0.3211 - val_accuracy: 0.87
Epoch 5/5
500/500 [==============================] - 201s 401ms/step - loss: 0.1784 - accuracy: 0.9314 - val_loss: 0.3522 - val_accuracy: 0.87
<keras.src.callbacks.History at 0x7f3e416bead0>
```

Model Evaluation

```
loss,accuracy=model.evaluate(X_test,Y_test)
print(f"Test loss:{loss}")
print(f"Test Accuracy:{accuracy}")
```

```
313/313 [==============================] - 21s 65ms/step - loss: 0.3324 - accuracy: 0.8791
Test loss:0.3323802947998047
Test Accuracy:0.8791000247001648
```

Building a Predictive System

```
def predict_sentiment(review):
  #tokenize and pad the review
  sequence=tokenizer.texts_to_sequences([review])
  padded_sequence=pad_sequences(sequence,maxlen=200)
  prediction=model.predict(padded_sequence)
  sentiment="positive" if prediction[0][0]>0.5 else "negative"
  return sentiment
```

```
#example
new_review="This movie was fantastic.I loved it."
sentiment=predict_sentiment(new_review)
print(f"The sentiment of the review is:{sentiment}")
```

```
1/1 [==============================] - 0s 281ms/step
The sentiment of the review is:positive
```

```
#example
new_review="This movie was terrible.I hated it."
sentiment=predict_sentiment(new_review)
print(f"The sentiment of the review is:{sentiment}")
```

```
1/1 [==============================] - 0s 73ms/step
The sentiment of the review is:negative
```