1. Importing dependencies:

In [ ]:
```
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, load_model
from tensorflow.keras.layers import Dense, Embedding, LSTM
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

2. Data Collection (Kagle API):

In [ ]:
```
kaggle_dic = json.load(open('kaggle.json'))
```

In [ ]:
```
kaggle_dic.keys()
```

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

A- Setup kaggle api as environement variables:

In [ ]:
```
os.environ["KAGGLE_USERNAME"] = kaggle_dic["username"]
os.environ["KAGGLE_KEY"] = kaggle_dic["key"]
```

B- Loading the dataset:

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

```
Dataset URL: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-mov
ie-reviews
License(s): other
imdb-dataset-of-50k-movie-reviews.zip: Skipping, found more recently modified local
copy (use --force to force download)
```

C- Unzib the dataset file:

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

3. Analyze the data:

In [ ]:
```
data = pd.read_csv('IMDB Dataset.csv')
```

A- Dimmensions of the data:

In [ ]:
```
data.shape
```

Out[ ]: (50000, 2)

B- Head of the data:

In [ ]:
```
data.head()
```

Out[ ]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. 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

C- Distrubutions of the data:

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

Out[ ]:
```
sentiment
positive    25000
negative    25000
Name: count, dtype: int64
```

D- Enconding our data:

In [ ]:
```
data.replace({"sentiment": {"positive": 1 , "negative" : 0}} ,inplace=True )
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_32096\4100101747.py:1: FutureWarning: Downc
asting behavior in `replace` is deprecated and will be removed in a future version.
To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To o
pt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', Tru
e)`
  data.replace({"sentiment": {"positive": 1 , "negative" : 0}} ,inplace=True )
```

In [ ]:
```
data.head()
```

Out[ ]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. 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

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

Out[ ]:
```
sentiment
1    25000
0    25000
Name: count, dtype: int64
```

E- Train test Split:

In [ ]:
```
train_data, test_data = train_test_split(data,test_size=0.2, random_state=42)
```

In [ ]:
```
print(train_data.shape)
print(test_data.shape)
```

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

4. Preprocessing the data:

A- Tokenize text data:

In [ ]:
```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(train_data["review"])
X_train = pad_sequences(tokenizer.texts_to_sequences(train_data["review"]), maxlen=
X_test = pad_sequences(tokenizer.texts_to_sequences(test_data["review"]), maxlen=20
```

In [ ]:
```
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]]
```

In [ ]:
```
X_test
```

Out[ ]:
```
array([[   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]])
```

B- Assigning the labels:

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

In [ ]:
```
Y_train
```

Out[ ]:
```
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
```

In [ ]:
```
Y_test
```

Out[ ]:
```
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
```

5. Neural Network LSTM (Long Short-Term Memory):

A- Build the model:

In [ ]:
```
#Initialisation du Modèle Séquentiel
model = Sequential()
#Couche d'embedding qui convertit les indices de mots en vecteur de dim finie
#On a 5000 mot , chat mot representer par un vecteur de 120 dimmenssion, chaque ent
model.add(Embedding(input_dim= 5000, output_dim=120,input_length=200))
#Chouche LSTM qui  traite les séquences de vecteurs produits par la couche d'Embedd
#128: dim de l'espcae de sortie
#Abandon aléatoire de 20%  aux unités de sortie de la couche pendant l'entraînement
#Abandon aléatoire de 20%  aux cnx recurrentes
model.add(LSTM(64,dropout=0.2, recurrent_dropout=0.2))
#Cette couche est une couche pleinement connectée qui suit la couche LSTM
#1: un seul unite de sortie
model.add(Dense(1, activation="sigmoid"))
```

```
WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-
packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Plea
se use tf.compat.v1.get_default_graph instead.
```

B- Information about the model:

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 120)          600000

 lstm (LSTM)                 (None, 64)                47360

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 647425 (2.47 MB)
Trainable params: 647425 (2.47 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

C- Compile the model:

In [ ]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

D- Training the model:

In [ ]:
```
model.fit(X_train,Y_train,epochs=5, batch_size=32, validation_split=0.2)
```

```
Epoch 1/5
WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

1000/1000 [==============================] - 129s 124ms/step - loss: 0.3816 - accuracy: 0.8290 - val_loss: 0.2933 - val_accuracy: 0.8781
Epoch 2/5
1000/1000 [==============================] - 115s 115ms/step - loss: 0.2636 - accuracy: 0.8956 - val_loss: 0.2925 - val_accuracy: 0.8801
Epoch 3/5
1000/1000 [==============================] - 117s 117ms/step - loss: 0.2131 - accuracy: 0.9169 - val_loss: 0.3102 - val_accuracy: 0.8774
Epoch 4/5
1000/1000 [==============================] - 118s 118ms/step - loss: 0.1827 - accuracy: 0.9299 - val_loss: 0.3977 - val_accuracy: 0.8605
Epoch 5/5
1000/1000 [==============================] - 116s 116ms/step - loss: 0.1515 - accuracy: 0.9420 - val_loss: 0.3390 - val_accuracy: 0.8823
```

Out[ ]:
```
<keras.src.callbacks.History at 0x236a41697c0>
```

6. Model Evaluation:

A- Accuracy and loss:

In [ ]:
```
loss, accuracy = model.evaluate(X_test,Y_test)
print(loss)
print(accuracy)
```

```
313/313 [==============================] - 10s 28ms/step - loss: 0.3312 - accuracy:
0.8802
0.3312399983406067
0.8802000284194946
```

B- Saving the model:

In [ ]:
```
model.save('sentiment_analysis_model.h5')
```

```
c:\Users\HP\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\src\engin
e\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `mode
l.save()`. This file format is considered legacy. We recommend using instead the nat
ive Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

7. Building a predeicitve systeme:

A- Loading the model

In [ ]:
```
model = load_model('sentiment_analysis_model.h5')
```

A- Function of prediction sentiments:

In [ ]:
```
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
```

B- Example!

In [ ]:
```
review = "Bad movie"
sentiment = predict_sentiment(review)
print(sentiment)
```

```
1/1 [==============================] - 0s 447ms/step
negative
```

In [ ]:
```
new_review = "This movie was ok but not that good."
sentiment = predict_sentiment(new_review)
print(f"The sentiment of the review is: {sentiment}")
```

```
1/1 [==============================] - 0s 67ms/step
The sentiment of the review is: negative
```

In [ ]:
```
review = "I like this movie wowww"
sentiment = predict_sentiment(review)
print(sentiment)
```

```
1/1 [==============================] - 0s 67ms/step
positive
```