first importing required **libraries**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


import nltk
nltk.download("punkt")
nltk.download("wordnet")
nltk.download("stopwords")

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence #unique id

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN, Dropout, Embedding, LSTM
import warnings
warnings.filterwarnings("ignore")
```

```
    [nltk_data] Downloading package punkt to /root/nltk_data...
                                        already up-to-date!
                                        wordnet to /root/nltk_data...
                                        s already up-to-date!
    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Package stopwords is already up-to-date!
```


uploading dataset

```
df = pd.read_csv("IMDB.csv")
df
```

	#	rating	title	username	review-date	review-text	actions-helpful	Type
0	1	10	One of the finest TV series coming from India.	authentic_writer	15-May-20	I was hooked to it from the very first episode...	66 out of 112 found this helpful.	Positive
1	2	10	Fantastic	ansegal	15-May-20	Spell bound I was and for a change the police ...	63 out of 110 found this helpful.	Positive
2	3	10	Prime does it again	abhisheksarkar901	16-May-20	Avoid watching it if you are disassociated wit...	9 out of 13 found this helpful.	Positive
3	4	10	One of finest of 2020	aspnishanth-81860	15-May-20	When u start watch it...you will binge watch i...	40 out of 77 found this helpful.	Positive
4	5	9	Worth Binge Watching	rahulreeves	16-May-20	Great story, content and justice done by the c...	7 out of 10 found this helpful.	Positive
...	...	...	...	...	...	...	...	...
370	371	9	Realistic	akjhacipet	16-May-20	Whole series looks very much real & story is c...	3 out of 6 found this helpful.	Positive
371	372	10	Quite hard hitting	KebbyPro	16-May-20	This series is not for the faint hearts. The m...	3 out of 6 found this helpful.	Positive
372	373	10	Mr Ahlawat You are Simply Awesome	dysondarryl	16-May-20	One of the best from Indian Series.... Raw Har...	3 out of 6 found this helpful.	Positive
373	374	10	You cant stop watching it	harmeet-mehta	16-May-20	Undoubtedly, till May 2020, Paatal Lok is the ...	3 out of 6 found this helpful.	Positive
374	375	10	After longtime	bua-raha	16-May-20	No unnecessary sleazy scenes, top of the actin...	3 out of 6 found this helpful.	Positive

375 rows × 8 columns

```
df.head()
```

	#	rating	title	username	review-date	review-text	actions-helpful	Type
0	1	10	One of the finest TV series coming from India.	authentic_writer	15-May-20	I was hooked to it from the very first episode...	66 out of 112 found this helpful.	Positive
1	2	10	Fantastic	ansegal	15-May-20	Spell bound I was and for a change the police ...	63 out of 110 found this helpful.	Positive
2	3	10	Prime does it again	abhisheksarkar901	16-May-20	Avoid watching it if you are disassociated wit...	9 out of 13 found this helpful.	Positive

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 375 entries, 0 to 374
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   #                375 non-null    int64 
 1   rating           375 non-null    int64 
 2   title            375 non-null    object
 3   username         375 non-null    object
 4   review-date      375 non-null    object
 5   review-text      375 non-null    object
 6   actions-helpful  375 non-null    object
 7   Type             375 non-null    object
dtypes: int64(2), object(6)
memory usage: 23.6+ KB
```



```
df["Type"].value_counts()
```

```
Positive    274
Negative    101
Name: Type, dtype: int64
```

```
df.head()
```

	#	rating	title	username	review-date	review-text	actions-helpful	Type
0	1	10	One of the finest TV series coming from India.	authentic_writer	15-May-20	I was hooked to it from the very first episode...	66 out of 112 found this helpful.	Positive
1	2	10	Fantastic	ansegal	15-May-20	Spell bound I was and for a change the police ...	63 out of 110 found this helpful.	Positive
2	3	10	Prime does it again	abhisheksarkar901	16-May-20	Avoid watching it if you are disassociated wit...	9 out of 13 found this helpful.	Positive
3	4	10	One of finest of 2020	aspnishanth-81860	15-May-20	When u start watch it...you will binge watch i...	40 out of 77 found this helpful.	Positive
4	5	9	Worth Binge Watching	rahulreeves	16-May-20	Great story, content and justice done by the c...	7 out of 10 found this helpful.	Positive

checking null values

```
df.isnull().sum()
```

```
#                0
rating           0
title            0
username         0
review-date      0
review-text      0
actions-helpful  0
Type             0
dtype: int64
```

creating a funtion to clean the reviews all at once

```
def cleantext(text):
  tokens = word_tokenize(text.lower())
  ftoken = [t for t in tokens if(t.isalpha())]
  stop = stopwords.words("english")
  ctoken = [t for t in ftoken if(t not in stop)]
  lemma = WordNetLemmatizer()
  ltoken = [lemma.lemmatize(t) for t in ctoken]
  return " ".join(ltoken)
```

applying the function

```
df["clean_review"]=df["review-text"].apply(cleantext)
```

```
df.head()
```

	#	rating	title	username	review-date	review-text	actions-helpful	Type	clean_review
0	1	10	One of the finest TV series coming from India.	authentic_writer	15-May-20	I was hooked to it from the very first episode...	66 out of 112 found this helpful.	Positive	hooked first episode main character true repre...
1	2	10	Fantastic	ansegal	15-May-20	Spell bound I was and for a change the police ...	63 out of 110 found this helpful.	Positive	spell bound change police seems real courtysid...
2	3	10	Prime does it again	abhisheksarkar901	16-May-20	Avoid watching it if you are disassociated wit...	9 out of 13 found this helpful.	Positive	avoid watching disassociated gruesome reality ...
3	4	10	One of finest of 2020	aspnishanth-81860	15-May-20	When u start watch it...you will binge watch i...	40 out of 77 found this helpful.	Positive	u start watch binge watch start ask season wan...
4	5	9	Worth Binge	rahulreeves	16-May-	Great story, content and justice done by the	7 out of 10 found this	Positive	great story content justice done character

spliting the data into x and y

Saving...

```
x = df["clean_review"]
y = df.Type.replace({'Negative':0,'Positive':1})

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.3, random_state=1)
```

now getting the sentence length

```
sentlen = []

for sent in df["clean_review"]:
  sentlen.append(len(word_tokenize(sent)))

df["SentLen"] = sentlen
df.head()
```

	#	rating	title	username	review-date	review-text	actions-helpful	Type	clean_review	SentLen
0	1	10	One of the finest TV series coming from India.	authentic_writer	15-May-20	I was hooked to it from the very first episode...	66 out of 112 found this helpful.	Positive	hooked first episode main character true repre...	27
1	2	10	Fantastic	ansegal	15-May-20	Spell bound I was and for a change the police ...	63 out of 110 found this helpful.	Positive	spell bound change police seems real courtysid...	35
2	3	10	Prime does it again	abhisheksarkar901	16-May-20	Avoid watching it if you are disassociated wit...	9 out of 13 found this helpful.	Positive	avoid watching disassociated gruesome reality ...	20
3	4	10	One of finest of 2020	aspnishanth-81860	15-May-20	When u start watch it...you will binge watch i...	40 out of 77 found this helpful.	Positive	u start watch binge watch start ask season wan...	10
4	5	9	Worth Binge	rahulreeves	16-May-	Great story, content and justice done by the	7 out of 10 found this	Positive	great story content justice done character	

```
max(sentlen)
```

```
386
```

maximum length of a review is 386

```
np.quantile(sentlen, 0.95)
```

```
75.0
```

here the 95% percentile of the data has 75 length of there sentence

```
max_len = np.quantile(sentlen, 0.95)
```

giving token to every characters in the dataset as per there appearence

```
tok = Tokenizer(char_level=False, split=" ")
#char_level if True, every character will be treated as a token.

tok.fit_on_texts(xtrain)
tok.index_word
```

```
      944: 'called',
      945: 'urban',
      946: 'arguably',
      947: 'follow',
      948: 'avoided',
      949: 'entertainer',
      950: 'underrated',
      951: 'viny',
      952: 'bind',
      953: 'breath',
      954: 'fresh',
      955: 'reprsents',
      956: 'blood',
      957: 'thirsty',
      958: 'remeber',
      959: 'addresing',
      960: 'hassil',
      961: 'essential',
      962: 'regret',
      963: 'hole',
      964: 'glued',
```



```
      968: 'fav',
      969: 'needed',
      970: 'sufficient',
      971: 'false',
      972: 'metaphor',
      973: 'symbolism',
      974: 'paatallok',
      975: 'dhartilok',
      976: 'enrooted',
      977: 'killing',
      978: 'total',
      979: 'advantage',
      980: 'paataal',
      981: 'dharti',
      982: 'ansari',
      983: 'cliche',
      984: 'masala',
      985: 'mixed',
      986: 'patallok',
      987: 'rock',
      988: 'white',
      989: 'safforn',
      990: 'disgusting',
      991: 'excellent',
      992: 'belief',
      993: 'firstly',
      994: 'congratulate',
      995: 'crew',
      996: 'producing',
      997: 'proved',
      998: 'technicality',
      999: 'graphic',
      1000: 'visuals',
      ...}
```

calculating total number of unique words

```
vocab_len = len(tok.index_word)
vocab_len
```

```
     1662
```

applying the token numbers to the reviews in the dataset

```
seqtrain = tok.texts_to_sequences(xtrain) #step1
seqtrain
```

```
        1031,
        410,
        218,
        1032,
        18,
        60,
        1,
        1033,
        379,
        34,
        1034,
        1035,
        1036,
        1037,
        72,
        7,
        8,
        281,
        100,
        44,
        1038,
        299,
        230,
        33],
       [90, 363, 240, 712, 241, 26, 2, 364, 713, 1],
       [16,
        10,
        262,
        14,
        92,
        23,
        4,
```



```
        826,
        1,
        156,
        69,
        599,
        186,
        129,
        174,
        342,
        279,
        53,
        827,
        828,
        829],
       [226, 40, 1, 29, 159, 5, 255, 36, 5, 10]]
```

now applying the sequence as per the max length and Tokenizing it

```
seqmattrain = sequence.pad_sequences(seqtrain, maxlen= int(max_len)) #step2
seqmattrain
```

```
array([[  0,   0,   0, ..., 364, 713,   1],
       [  0,   0,   0, ..., 216,  60,   3],
       [  0,   0,   0, ...,  77, 189, 134],
       ...,
       [  0,   0,   0, ..., 364, 713,   1],
       [  0,   0,   0, ..., 827, 828, 829],
       [  0,   0,   0, ...,  36,   5,  10]], dtype=int32)
```

saving it in a variable to perform training and testing

```
seqtest = tok.texts_to_sequences(xtest)
seqmattest = sequence.pad_sequences(seqtest, maxlen=int(max_len))
```

```
vocab_len
```

```
1662
```

Building the RNN Model

```
rnn = Sequential()

rnn.add(Embedding(vocab_len+1,90,input_length=int(max_len), mask_zero=True))
rnn.add(SimpleRNN(units=32, activation="tanh"))
rnn.add(Dense(units=32, activation="relu"))
rnn.add(Dropout(0.2))

rnn.add(Dense(units=1, activation="sigmoid"))

rnn.compile(optimizer="adam", loss="binary_crossentropy")

rnn.fit(seqmattrain, ytrain, batch_size=50, epochs=30)
```

```
ypred = rnn.predict(seqmattest)

ypred = ypred>0.5
```

```
6/6 [==============================] - 0s 33ms/step - loss: 0.4995
Epoch 3/30
6/6 [==============================] - 0s 33ms/step - loss: 0.3529
Epoch 4/30
6/6 [==============================] - 0s 38ms/step - loss: 0.2332
Epoch 5/30
6/6 [==============================] - 0s 31ms/step - loss: 0.1430
Epoch 6/30
6/6 [==============================] - 0s 34ms/step - loss: 0.0959
Epoch 7/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0694
Epoch 8/30
6/6 [==============================] - 0s 33ms/step - loss: 0.0426
Epoch 9/30
6/6 [==============================] - 0s 35ms/step - loss: 0.0306
Epoch 10/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0223
Epoch 11/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0163
Epoch 12/30
6/6 [==============================] - 0s 33ms/step - loss: 0.0148
Epoch 13/30
6/6 [==============================] - 0s 34ms/step - loss: 0.0130
Epoch 14/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0088
```



```
6/6 [==============================] - 0s 33ms/step - loss: 0.0078
6/6 [==============================] - 0s 34ms/step - loss: 0.0067
Epoch 17/30
6/6 [==============================] - 0s 33ms/step - loss: 0.0050
Epoch 18/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0055
Epoch 19/30
6/6 [==============================] - 0s 33ms/step - loss: 0.0046
Epoch 20/30
6/6 [==============================] - 0s 35ms/step - loss: 0.0037
Epoch 21/30
6/6 [==============================] - 0s 34ms/step - loss: 0.0039
Epoch 22/30
6/6 [==============================] - 0s 34ms/step - loss: 0.0033
Epoch 23/30
6/6 [==============================] - 0s 37ms/step - loss: 0.0027
Epoch 24/30
6/6 [==============================] - 0s 33ms/step - loss: 0.0026
Epoch 25/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0031
Epoch 26/30
6/6 [==============================] - 0s 33ms/step - loss: 0.0023
Epoch 27/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0021
Epoch 28/30
6/6 [==============================] - 0s 37ms/step - loss: 0.0024
Epoch 29/30
6/6 [==============================] - 0s 33ms/step - loss: 0.0021
Epoch 30/30
6/6 [==============================] - 0s 32ms/step - loss: 0.0021
4/4 [==============================] - 0s 8ms/step
```

checking the Accuracy

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
                  precision    recall  f1-score   support

               0       0.62      0.90      0.74        31
               1       0.96      0.79      0.87        82

        accuracy                           0.82       113
       macro avg       0.79      0.85      0.80       113
    weighted avg       0.86      0.82      0.83       113
```

```
#======================
```

now BUilding and applying the LSTM Model

```
lstm = Sequential()

lstm.add(Embedding(vocab_len+1,90,input_length=int(max_len), mask_zero=True))
lstm.add(LSTM(units=32, activation="tanh"))
lstm.add(Dense(units=32, activation="relu"))
```

```
lstm.add(Dropout(0.2))

lstm.add(Dense(units=1, activation="sigmoid"))

lstm.compile(optimizer="adam", loss="binary_crossentropy")

lstm.fit(seqmattrain, ytrain, batch_size=50, epochs=30)

ypred = lstm.predict(seqmattest)

ypred = ypred>0.5
```

```
6/6 [==============================] - 0s 58ms/step - loss: 0.6621
Epoch 3/30
6/6 [==============================] - 0s 60ms/step - loss: 0.6200
Epoch 4/30
6/6 [==============================] - 0s 61ms/step - loss: 0.5298
Epoch 5/30
6/6 [==============================] - 1s 93ms/step - loss: 0.4133
Epoch 6/30
6/6 [==============================] - 1s 106ms/step - loss: 0.3191
Epoch 7/30
6/6 [==============================] - 1s 138ms/step - loss: 0.2529
Epoch 8/30
6/6 [==============================] - 1s 123ms/step - loss: 0.1941
Epoch 9/30
6/6 [==============================] - 1s 128ms/step - loss: 0.1424
Epoch 10/30
6/6 [==============================] - 1s 138ms/step - loss: 0.1065
```

Saving...

```
6/6 [==============================] - 1s 106ms/step - loss: 0.0777
6/6 [==============================] - 1s 112ms/step - loss: 0.0516
Epoch 13/30
6/6 [==============================] - 1s 120ms/step - loss: 0.0334
Epoch 14/30
6/6 [==============================] - 1s 108ms/step - loss: 0.0201
Epoch 15/30
6/6 [==============================] - 1s 104ms/step - loss: 0.0170
Epoch 16/30
6/6 [==============================] - 1s 121ms/step - loss: 0.0098
Epoch 17/30
6/6 [==============================] - 1s 175ms/step - loss: 0.0078
Epoch 18/30
6/6 [==============================] - 1s 180ms/step - loss: 0.0069
Epoch 19/30
6/6 [==============================] - 1s 219ms/step - loss: 0.0058
Epoch 20/30
6/6 [==============================] - 1s 123ms/step - loss: 0.0040
Epoch 21/30
6/6 [==============================] - 1s 124ms/step - loss: 0.0038
Epoch 22/30
6/6 [==============================] - 1s 117ms/step - loss: 0.0031
Epoch 23/30
6/6 [==============================] - 1s 146ms/step - loss: 0.0038
Epoch 24/30
6/6 [==============================] - 1s 150ms/step - loss: 0.0029
Epoch 25/30
6/6 [==============================] - 1s 170ms/step - loss: 0.0028
Epoch 26/30
6/6 [==============================] - 1s 151ms/step - loss: 0.0024
Epoch 27/30
6/6 [==============================] - 1s 161ms/step - loss: 0.0020
Epoch 28/30
6/6 [==============================] - 1s 74ms/step - loss: 0.0019
Epoch 29/30
6/6 [==============================] - 0s 58ms/step - loss: 0.0019
Epoch 30/30
6/6 [==============================] - 0s 59ms/step - loss: 0.0015
4/4 [==============================] - 1s 13ms/step
```

accuracy after Lstm model

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        31
           1       1.00      0.96      0.98        82

    accuracy                           0.97       113
   macro avg       0.96      0.98      0.97       113
weighted avg       0.98      0.97      0.97       113
```

Hence The ACCURACY of the Model is better in LSTM compared to RNN