1)Download and upload the dataset in colab

```
!unzip '/content/spam.csv'
```

```
Archive:  /content/spam.csv
  End-of-central-directory signature not found.  Either this file is
  a zipfile, or it constitutes one disk of a multi-part archive.  In
  latter case the central directory and zipfile comment will be found
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of /content/spam.csv or
        /content/spam.csv.zip, and cannot find /content/spam.csv.ZIP,
```

2)Import the required library

```
import numpy as np
import pandas as pd
import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from keras.layers import Embedding
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras_preprocessing.sequence import pad_sequences
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

3)Import the required library

```
df = pd.read_csv('/content/spam.csv', encoding="ISO-8859-1")
```

```
df
```

	v1	v2	Unnamed: 2	Uni
0	ham	Go until jurong point, crazy.. Available only ...	NaN	
1	ham	Ok lar... Joking wif u oni...	NaN	
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	
3	ham	U dun say so early hor... U c already then say...	NaN	
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	
...	...	...	...	
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	
5570	ham	The guy did some bitching but I acted like i'd...	NaN	
5571	ham	Rofl. Its true to its name	NaN	

5572 rows × 5 columns

```
data = df[['v1', 'v2']]
data
```

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...

```
ps = PorterStemmer()

for i in range(0, 5572):
  review = data['v2'][i]
  review = re.sub('[^a-zA-Z]',' ', review)
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if word not in set(stopwords.
  review = ' '.join(review)
  data['v2'][i] = review
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: Set
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
  # Remove the CWD from sys.path while we load stuff.
```

```
data
```

v1 v2

```
Max = 50000
Max_seq = 250
emb = 100

tokenizer = Tokenizer(num_words = Max)
tokenizer.fit_on_texts(data['v2'].values)
word_index = tokenizer.word_index

x = tokenizer.texts_to_sequences(data['v2'].values)
x = pad_sequences(x, maxlen = Max_seq)

y = pd.get_dummies(data['v1']).values

print(x.shape, y.shape)
```

```
(5572, 250) (5572, 2)
```

5571 ham rofl true name

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y)
print(xtrain.shape, ytrain.shape)
print(xtest.shape, ytest.shape)
```

```
(4179, 250) (4179, 2)
(1393, 250) (1393, 2)
```

```
xtrain.reshape(4179, 250, 1)
ytrain.reshape(4179, 2, 1)
xtest.reshape(1393, 250, 1)
ytest.reshape(1393, 2, 1)
```

```
array([[[1],
        [0]],

       [[1],
        [0]],

       [[1],
        [0]],

       ...,

       [[1],
        [0]],
```

```
       [[1],
        [0]],

       [[1],
        [0]]], dtype=uint8)
```

4)Create Model

```
model = Sequential()
```

5)Add Layers

```
model.add(Embedding(Max, emb, input_length = x.shape[1]))
model.add(LSTM(100))
model.add(Dense(2, activation = 'relu'))
```

6)Compile Model

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

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 250, 100)          5000000

 lstm (LSTM)                 (None, 100)               80400

 dense (Dense)               (None, 2)                 202

=================================================================
Total params: 5,080,602
Trainable params: 5,080,602
Non-trainable params: 0
_________________________________________________________________
```

7)Fit the model

```
model.fit(xtrain,ytrain,epochs=10)
```

```
Epoch 1/10
131/131 [==============================] - 33s 231ms/step - loss: 0.6
Epoch 2/10
131/131 [==============================] - 31s 234ms/step - loss: 0.6
Epoch 3/10
131/131 [==============================] - 31s 235ms/step - loss: 0.6
Epoch 4/10
131/131 [==============================] - 30s 230ms/step - loss: 0.6
Epoch 5/10
131/131 [==============================] - 31s 237ms/step - loss: 0.6
Epoch 6/10
131/131 [==============================] - 30s 231ms/step - loss: 0.6
Epoch 7/10
131/131 [==============================] - 30s 230ms/step - loss: 8.5
Epoch 8/10
131/131 [==============================] - 30s 231ms/step - loss: 6.3
Epoch 9/10
131/131 [==============================] - 30s 231ms/step - loss: 4.9
Epoch 10/10
131/131 [==============================] - 30s 231ms/step - loss: 4.6
<keras.callbacks.History at 0x7fb7167ba990>
```

8)Save the model

```
model.save('MailChecker.h5')
```

9)Test the model

```
op = ['ham', 'spam']

def text_processing(text):
  review = re.sub('[^a-zA-Z]',' ', text)
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if word not in set(stopwords.
  review = ' '.join(review)
  return review


# Testing 1
text =  '''Welcome,Hello world'''
```

```
text = text_processing(text)
seq = tokenizer.texts_to_sequences([text])
padded = pad_sequences(seq, maxlen = Max_seq)
pred = model.predict(padded)

print(pred, op[np.argmax(pred)])
```

```
1/1 [==============================] - 0s 487ms/step
[[0.8593883  0.16747844]] ham
```

```
# Testing 2
text =  '''Pleasure to have you,have a luxary stay'''

text = text_processing(text)
seq = tokenizer.texts_to_sequences([text])
padded = pad_sequences(seq, maxlen = Max_seq)
pred = model.predict(padded)

print(pred, op[np.argmax(pred)])
```

```
1/1 [==============================] - 0s 34ms/step
[[0.92849386 0.        ]] ham
```

```
# Testing 3

text =  '''Hurry!!
           you have won it'''

text = text_processing(text)
seq = tokenizer.texts_to_sequences([text])
padded = pad_sequences(seq, maxlen = Max_seq)
pred = model.predict(padded)

print(pred, op[np.argmax(pred)])
```

```
1/1 [==============================] - 0s 35ms/step
[[0.89900625 0.07140643]] ham
```

```
# Testing 4

text =  '''Have a blissful day'''

text = text_processing(text)
seq = tokenizer.texts_to_sequences([text])
```

```
seq = tokenizer.texts_to_sequences([text])
padded = pad_sequences(seq, maxlen = Max_seq)
pred = model.predict(padded)

print(pred, op[np.argmax(pred)])
```

```
1/1 [==============================] - 0s 36ms/step
[[0.957064 0.      ]] ham
```

Colab paid products - Cancel contracts here

✓ 0s completed at 9:53 PM