

## 1.Import requiried library

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input,
Embedding
from keras.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import to_categorical
from keras.models import load_model
```

```
import csv
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
```

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

## 2.Read dataset and do preprocessing

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df =
pd.read_csv('/content/drive/MyDrive/spam.csv',delimiter=',',encoding='
latin-1')
df.head()
```

	v1	v2 Unnamed: 2
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

```

```

      Unnamed: 3 Unnamed: 4
0           NaN          NaN
1           NaN          NaN
2           NaN          NaN
3           NaN          NaN
4           NaN          NaN

```

```

df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
#dropping unwanted columns
df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  -
0    v1      5572 non-null     object
1    v2      5572 non-null     object
dtypes: object(2)
memory usage: 87.2+ KB

```

```

#Count of Spam and Ham values
df.groupby(['v1']).size()

```

```

v1
ham      4825
spam      747
dtype: int64

```

```

#Label Encoding target column

```

```

X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)

```

```

# Test and train split

```

```

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)

```

```

# Tokenisation function

```

```

max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = sequence.pad_sequences(sequences,maxlen=max_len)

```

### 3.Create Model 4.Add layers (LSTM ,Dense-(Hidden Layers),Ouput)

```
#creating LSTM model
inputs = Input(name='InputLayer',shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(64)(layer)
layer = Dense(256,name='FullyConnectedLayer1')(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1,name='OutputLayer')(layer)
layer = Activation('sigmoid')(layer)
```

### 5. Compile the model

```
model = Model(inputs=inputs,outputs=layer)
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=[
'accuracy'])
```

Model: "model"

Layer (type)	Output Shape	Param #
InputLayer (InputLayer)	[(None, 150)]	0
embedding (Embedding)	(None, 150, 50)	50000
lstm (LSTM)	(None, 64)	29440
FullyConnectedLayer1 (Dense )	(None, 256)	16640
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
OutputLayer (Dense)	(None, 1)	257
activation_1 (Activation)	(None, 1)	0
Total params: 96,337		
Trainable params: 96,337		
Non-trainable params: 0		

### 6. Fit the model

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
validation_split=0.2)
```

```

Epoch 1/10
30/30 [=====] - 8s 263ms/step - loss: 0.0035
- accuracy: 0.9995 - val_loss: 0.1122 - val_accuracy: 0.9863
Epoch 2/10
30/30 [=====] - 13s 454ms/step - loss: 0.0026
- accuracy: 0.9995 - val_loss: 0.1018 - val_accuracy: 0.9873
Epoch 3/10
30/30 [=====] - 14s 468ms/step - loss: 0.0026
- accuracy: 0.9992 - val_loss: 0.0911 - val_accuracy: 0.9852
Epoch 4/10
30/30 [=====] - 15s 493ms/step - loss: 0.0023
- accuracy: 0.9995 - val_loss: 0.1240 - val_accuracy: 0.9852
Epoch 5/10
30/30 [=====] - 10s 349ms/step - loss: 0.0015
- accuracy: 0.9995 - val_loss: 0.1336 - val_accuracy: 0.9863
Epoch 6/10
30/30 [=====] - 7s 249ms/step - loss: 0.0026
- accuracy: 0.9992 - val_loss: 0.1339 - val_accuracy: 0.9873
Epoch 7/10
30/30 [=====] - 9s 289ms/step - loss:
3.0076e-04 - accuracy: 0.9997 - val_loss: 0.1313 - val_accuracy:
0.9873
Epoch 8/10
30/30 [=====] - 8s 255ms/step - loss:
4.5712e-04 - accuracy: 0.9997 - val_loss: 0.1547 - val_accuracy:
0.9873
Epoch 9/10
30/30 [=====] - 8s 253ms/step - loss:
1.8049e-04 - accuracy: 1.0000 - val_loss: 0.1490 - val_accuracy:
0.9863
Epoch 10/10
30/30 [=====] - 11s 366ms/step - loss:
4.6702e-05 - accuracy: 1.0000 - val_loss: 0.1521 - val_accuracy:
0.9873

```

```
<keras.callbacks.History at 0x7f284144c9d0>
```

## 7. Save the model

```
model.save("model_1")
```

```

WARNING:absl:Function `_wrapped_model` contains input name(s)
InputLayer with unsupported characters which will be renamed to
inputlayer in the SavedModel.
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses while saving
(showing 2 of 2). These functions will not be directly callable after
loading.

```

## 8. Test the Model

[illegible]