1. Download the dataset

2. Import requied 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]   Package stopwords is already up-to-date!
```

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

3. Read dataset and do preprocessing

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

To see first 5 rows in dataset

```
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
```

Dropping unwanted columns

```
df.drop(['Unnamed: 2','Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
```

To get summary of the dataset

```
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
```

To get 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)
```

Spliting the data into training set and testing set

```
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)
```

4. Creating a model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense
```

5. Adding layers

```
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)

model = Model(inputs=inputs,outputs=layer)
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 InputLayer (InputLayer)     [(None, 150)]             0

 embedding_1 (Embedding)     (None, 150, 50)           50000

 lstm_1 (LSTM)               (None, 64)                29440

 FullyConnectedLayer1 (Dense  (None, 256)              16640
 )

 activation_2 (Activation)   (None, 256)               0

 dropout_1 (Dropout)         (None, 256)               0

 OutputLayer (Dense)         (None, 1)                 257
 activation_3 (Activation)   (None, 1)                 0

=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________________________________________
```

6. Compile the model

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

7. Fit the model

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

```
Epoch 1/10
30/30 [==============================] - 13s 329ms/step - loss: 0.3208
- accuracy: 0.8809 - val_loss: 0.1565 - val_accuracy: 0.9768
Epoch 2/10
30/30 [==============================] - 10s 324ms/step - loss: 0.0800
- accuracy: 0.9805 - val_loss: 0.0696 - val_accuracy: 0.9810
Epoch 3/10
30/30 [==============================] - 9s 296ms/step - loss: 0.0398
- accuracy: 0.9892 - val_loss: 0.0829 - val_accuracy: 0.9768
Epoch 4/10
30/30 [==============================] - 9s 303ms/step - loss: 0.0277
- accuracy: 0.9913 - val_loss: 0.0848 - val_accuracy: 0.9747
Epoch 5/10
30/30 [==============================] - 9s 305ms/step - loss: 0.0199
- accuracy: 0.9939 - val_loss: 0.0893 - val_accuracy: 0.9768
Epoch 6/10
30/30 [==============================] - 10s 344ms/step - loss: 0.0122
- accuracy: 0.9966 - val_loss: 0.0969 - val_accuracy: 0.9810
Epoch 7/10
30/30 [==============================] - 9s 302ms/step - loss: 0.0111
- accuracy: 0.9966 - val_loss: 0.1346 - val_accuracy: 0.9736
Epoch 8/10
30/30 [==============================] - 9s 299ms/step - loss: 0.0075
- accuracy: 0.9979 - val_loss: 0.1451 - val_accuracy: 0.9715
Epoch 9/10
30/30 [==============================] - 9s 305ms/step - loss: 0.0070
- accuracy: 0.9979 - val_loss: 0.1349 - val_accuracy: 0.9810
Epoch 10/10
30/30 [==============================] - 9s 304ms/step - loss: 0.0057
- accuracy: 0.9989 - val_loss: 0.1549 - val_accuracy: 0.9821
```

```
<keras.callbacks.History at 0x7f40ee615c50>
```

8. 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_1_layer_call_fn,
lstm_cell_1_layer_call_and_return_conditional_losses while saving
(showing 2 of 2). These functions will not be directly callable after
loading.
```

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix
=sequence.pad_sequences(test_sequences,maxlen=max_len)
accuracy = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy: {:0.3f}'.format(accuracy[1]))
```

```
27/27 [==============================] - 1s 26ms/step - loss: 0.1476 -
accuracy: 0.9809
Accuracy: 0.981
```

```
y_pred = model.predict(test_sequences_matrix)
print(y_pred[25:40].round(3))
```

```
27/27 [==============================] - 1s 26ms/step
[[0.   ]
 [0.   ]
 [0.998]
 [0.   ]
 [0.001]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.001]
 [1.   ]
 [0.   ]
 [1.   ]
 [0.   ]]
```

9. Test the model

```
print(Y_test[25:40])
```

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