Importing Required Libraries

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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.callbacks import EarlyStopping
from keras.utils import pad_sequences
%matplotlib inline
```

Read Dataset and Preprocessing

In [ ]:

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

Out[ ]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [ ]:

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

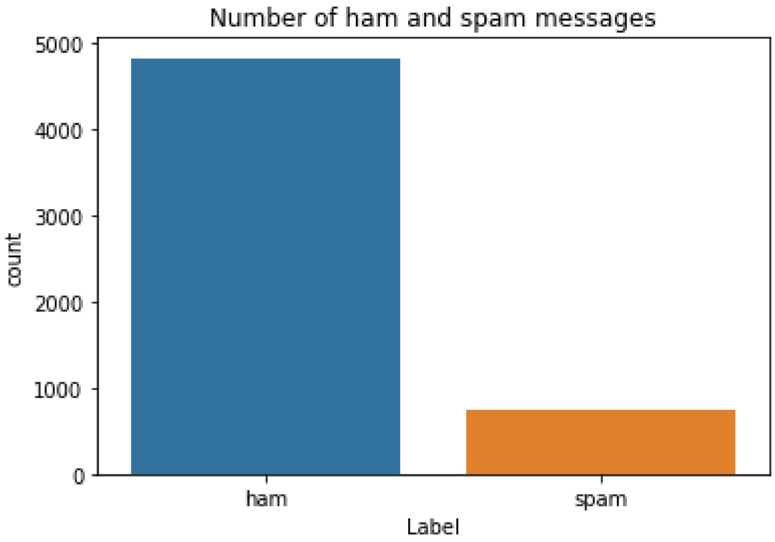
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [ ]:

```
sns.countplot(df.v1)
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
X = df.v2
Y = df.v1
```

```
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  FutureWarning
```



In [ ]:

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

In [ ]:

```
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 = pad_sequences(sequences,maxlen=max_len)
```

Create Model

In [ ]:

```
def RNN():
    inputs = Input(name='inputs',shape=[max_len])
    layer = Embedding(max_words,50,input_length=max_len)(inputs)
    layer = LSTM(64)(layer)
    layer = Dense(256,name='FC1')(layer)
    layer = Activation('relu')(layer)
    layer = Dropout(0.5)(layer)
    layer = Dense(1,name='out_layer')(layer)
    layer = Activation('sigmoid')(layer)
    model = Model(inputs=inputs,outputs=layer)
    return model
```

Adding LSTM Layers

In [ ]:

```
model = RNN()
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 inputs (InputLayer)         [(None, 150)]             0         
                                                                 
 embedding (Embedding)       (None, 150, 50)           50000     
                                                                 
 lstm (LSTM)                 (None, 64)                29440     
                                                                 
 FC1 (Dense)                 (None, 256)               16640     
                                                                 
 activation (Activation)     (None, 256)               0         
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 out_layer (Dense)           (None, 1)                 257       
                                                                 
 activation_1 (Activation)   (None, 1)                 0         
                                                                 
=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________________________________________
```

Compile The Model

In [ ]:

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

Fit The Model

In [ ]:

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
          validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta=
```

```
Epoch 1/10
30/30 [==============================] - 8s 279ms/step - loss: 0.0472 - accuracy: 0.9
876 - val_loss: 0.0598 - val_accuracy: 0.9842
Epoch 2/10
30/30 [==============================] - 8s 277ms/step - loss: 0.0367 - accuracy: 0.9
902 - val_loss: 0.0613 - val_accuracy: 0.9852
```

Out[ ]:

```
<keras.callbacks.History at 0x7f4c60cffe50>
```

Save The Model

In [ ]:

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

Test The Model

In [ ]:

```
test_sequences = tok.texts_to_sequences(X_test)
```

```
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
test_sequences_matrix
```

Out[ ]:

```
array([[  0,   0,   0, ...,  69,   9,   6],
       [  0,   0,   0, ...,  38,  17,   2],
       [  0,   0,   0, ..., 667,  52,  91],
       ...,
       [  0,   0,   0, ..., 586,  68, 361],
       [  0,   0,   0, ..., 535, 705,  65],
       [  0,   0,   0, ...,  12, 969, 407]], dtype=int32)
```

Accuracy Of The Model

In [ ]:

```
accr = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy:',accr[1])
print('Loss:',accr[0])
```

```
27/27 [==============================] - 1s 25ms/step - loss: 0.0303 - accuracy: 0.99
16
Accuracy: 0.9916267991065979
Loss: 0.03027874417603016
```